

AD-A138 876

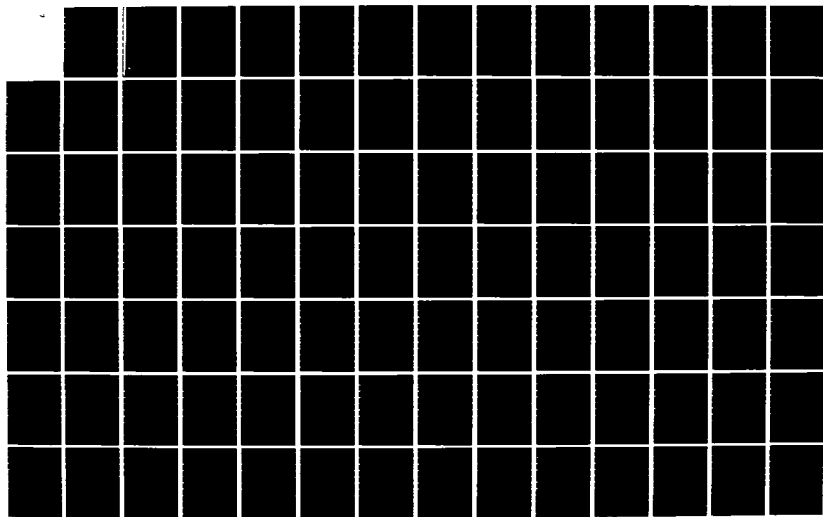
ADAPTIVE HYBRID PICTURE CODING(U) ARKANSAS UNIV
FAYETTEVILLE DEPT OF ELECTRICAL ENGINEERING
R A JONES ET AL. 30 NOV 83 AFOSR-TR-84-0142
AFOSR-82-0351

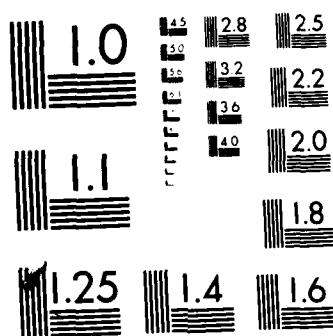
1/3

UNCLASSIFIED

F/G 12/1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

UNIVERSITY OF ARKANSAS

AD A138876

ADAPTIVE HYBRID PICTURE CODING

INTERIM SCIENTIFIC REPORT

by

Richard A. Jones
Principal Investigator

Mark K. Cook
Senior Investigator

Department of Electrical Engineering
University of Arkansas
Fayetteville, Arkansas 72701

November 30, 1983

DTIC

Prepared for the

**AIR FORCE OFFICE
OF SCIENTIFIC RESEARCH**

Under Grant No. AFOSR-82-0351

84 03 26 21

Approved for public release;
distribution unlimited.

DTIC FILE COPY

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 84-0142	2. GOVT ACCESSION NO. AD A138876	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADAPTIVE HYBRID PICTURE CODING		5. TYPE OF REPORT & PERIOD COVERED INTERIM REPORT 01 APR 1983 to 30 NOV 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard A. Jones, & Mark K. Cook		8. CONTRACT OR GRANT NUMBER(s) AFOSR-82-0351
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Arkansas Department of Electrical Engineering Fayetteville, AK 72701		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 6/102F 2305/B3
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research /NE Building 410 Bolling AFB, DC 20332		12. REPORT DATE Nov 30, 1983
		13. NUMBER OF PAGES 192
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The transmission of digital imagery has become a necessity in recent times. Systems such as communications and weather satellites, facsimile, remote control, and machine intelligence can and do make use of data compression techniques to reduce bandwidth and power consumption. Research on these techniques has led to one form of image data compression which achieves good image quality for intra-frame coding at low data rates. This technique is known as Adaptive Stochastic Picture Coding (ASPC) which consists of a one-dimensional unitary transform for -OVER-		

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

colum-wise decorrelation used in conjunction with Adaptive Differential Predictive Coding Modulation (ADPCM) for the row-wise decorrelation, followed by quantization to give the desired data compression. This system requires use of quantization techniques which limit system performance. Optimization of adaptive scalar quantizers and use of vector quantizers aid in the adaptation of the system to variations in the image statistics. This report represents a study of such quantizers in the ASPC system. By examining these quantization methods, it will be shown that it is vital that the proper quantizer be incorporated into the system to achieve a particular data rate at desired distortion levels.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

INTERIM SCIENTIFIC REPORT

Grant No. ~~82-0351~~ AFOSR-82-0351

ADAPTIVE HYBRID PICTURE CODING

Richard A. Jones
Principal Investigator

Mark K. Cook
Senior Investigator

University of Arkansas
Department of Electrical Engineering
Fayetteville, Arkansas 72701



30 November 1983

A1

Prepared for the Air Force Office of Scientific Research

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF FINAL REPORT TO DTIC
This report is classified as Unclassified and is
available to the public. DTIC Report No. 83-12.
Distribution is unlimited.
MATTHEW J. [illegible]
Chief, Technical Information Division

Table of Contents

Abstract	ii
List of Figuresiii
I. Introduction	1
II. Overview of Quantizer Theory	4
Background	4
Time Invariant Quantizers	6
Uniform Quantization	6
Non-Uniform Quantization	9
Companding	12
Time Varying Quantizers	14
Adaptive Quantizers	14
Coding	21
Vector Quantizers	23
Random Quantizers and Code Book Design	27
Lattice Quantizers	29
III. The ASPC System	33
DPCM Systems	33
Transformation Systems	37
Hybrid Systems	40
IV. Max-Lloyd Quantizer	46
V. Vector Quantizer Design	53
VI. System Configurations and Program Descriptions.	62
ASPC / Forward Adaptive Scalar Max	62
ASPC / Backward Adaptive Scalar Max	70
ASPC / Vector Quantization	74
ASPC / Scalar Max with Zeroing	79
Program Descriptions	81
VII. Results and Conclusions	86
Future Research	94
Bibliography	96
Appendix A: Pictures and Error Histograms100
Appendix B: Program Listings132

ABSTRACT

✓ The transmission of digital imagery has become a necessity in recent times. Systems such as communications and weather satellites, facsimile, remote control, and machine intelligence can and do make use of data compression techniques to reduce bandwidth and power consumption.

Research on these techniques has led to one form of image data compression which achieves good image quality for intraframe coding at low data rates. This technique is known as Adaptive Stochastic Picture Coding (ASPC) which consists of a one-dimensional unitary transform for column-wise decorrelation used in conjunction with Adaptive Differential Predictive Coding Modulation (ADPCM) for the row-wise decorrelation, followed by quantization to give the desired data compression. This system requires use of quantization techniques which limit system performance. Optimization of adaptive scalar quantizers and use of vector quantizers aid in the adaptation of the system to variations in the image statistics. This report represents a study of such quantizers in the ASPC system. By examining these quantization methods, it will be shown that it is vital that the proper quantizer be incorporated into the system to achieve a particular data rate at desired distortion levels.

✓

List of Figures

1. Classes of Quantizers	5
2. Uniform Quantizer	7
3. Scaled Uniform Quantizer	8
4. Quantizer Types	10
5. Companding	13
6. DPCM	15
7. Adaptive Quantizer	17
8. General Quantizer	18
9. Block Quantizer	24
10. Transform Coding System	25
11. Lattice Quantizer	30
12. Intra-frame DPCM	32
13. DPCM	36
14. 2-Dim Transform Coding	39
15. Hybrid Transform/DPCM Coding	41
16. Optimum Quantization	51
17. 2-Dim Vector Quantizer	54
18. ASPC Transmitter with Forward Adaptive Max	63
19. ASPC Receiver with Forward Adaptive Max	64
20. ASPC Transmitter with Backward Adaptive Max	72
21. ASPC Receiver with Backward Adaptive Max	73
22. Vector ASPC Transmitter	75
23. Vector ASPC Receiver	76
24. ASPC SNR Table	87
25. ASPC Predictor Performance	88
26. ASPC Quantizer Performance	91

CHAPTER ONE

INTRODUCTION

The purpose of the work reported here is to examine various quantization and coding methods for suitability as an integral element in an image data compression and transmission system -- specifically the Adaptive Stochastic Picture Coding (ASPC) system (sometimes also known as the Adaptive Hybrid Picture Coding system - AHPC). Pursuant to this aim, it was necessary to perform a qualitative analysis of system requirements and performance objectives. On the completion of this analysis, the results constituted a set of guidelines for the desired system operation.

The main criteria specified is the ability to transmit a picture with a good overall subjective reconstruction at low data rates. This is rephrased as a desire to transmit pictures at low data rates with an "acceptable" amount of distortion. As is expected, built in limitations of the system (ASPC in this case) prevent the achievement of the theoretical optimal performance. One can however, "fix" most of the system configuration to a set standard and study the effects of

replacing an element of the system (here the quantizer) with various experimental schemes. The observed results of the system's performance for each scheme are then evaluated in terms of meeting objective criteria.

Since the target of this report was the study of the effects of various quantizers on the AHPC system, the techniques of fundamental quantizer design were applied, and the different configurations of quantizers were incorporated into the system. Extensive computer simulation of each quantizer implementation was made in order that the system performance could be observed for a cross-comparison of quantizer schemes.

Since very little of the visual process is understood to date, many image processing systems are modeled after communications and speech processing systems. Although image and speech systems exemplify two different signal processing applications, they both exhibit similarities in the type of signals they must deal with and the ways in which these signals are dealt. Due to the similarities of the design of ASPC to speech processing systems, the analysis and design of several of the quantization methods were paralleled to those used in speech processing applications.

The following chapters will describe the theory and outcomes of this quantizer study, beginning with the basic fundamentals of quantization theory appearing in Chapter

Two, followed by a description of the ASPC system in Chapter Three. Chapter Four presents the derivation of the Max-Lloyd concept for scalar quantization. Chapter Five gives the derivation and design algorithm for vector quantization. Chapter Six gives detailed explanations of the various system configurations and program simulation descriptions. Chapter Seven summarizes the results and conclusions of the report including performance evaluations, data rate calculations, and thoughts about future research projects incorporating these findings. Appendix A gives the supporting photographs of the image reconstruction as well as all error histograms between the original image and received image as support for conclusions drawn in Chapter Seven. Appendix B gives the program flow charts and listings.

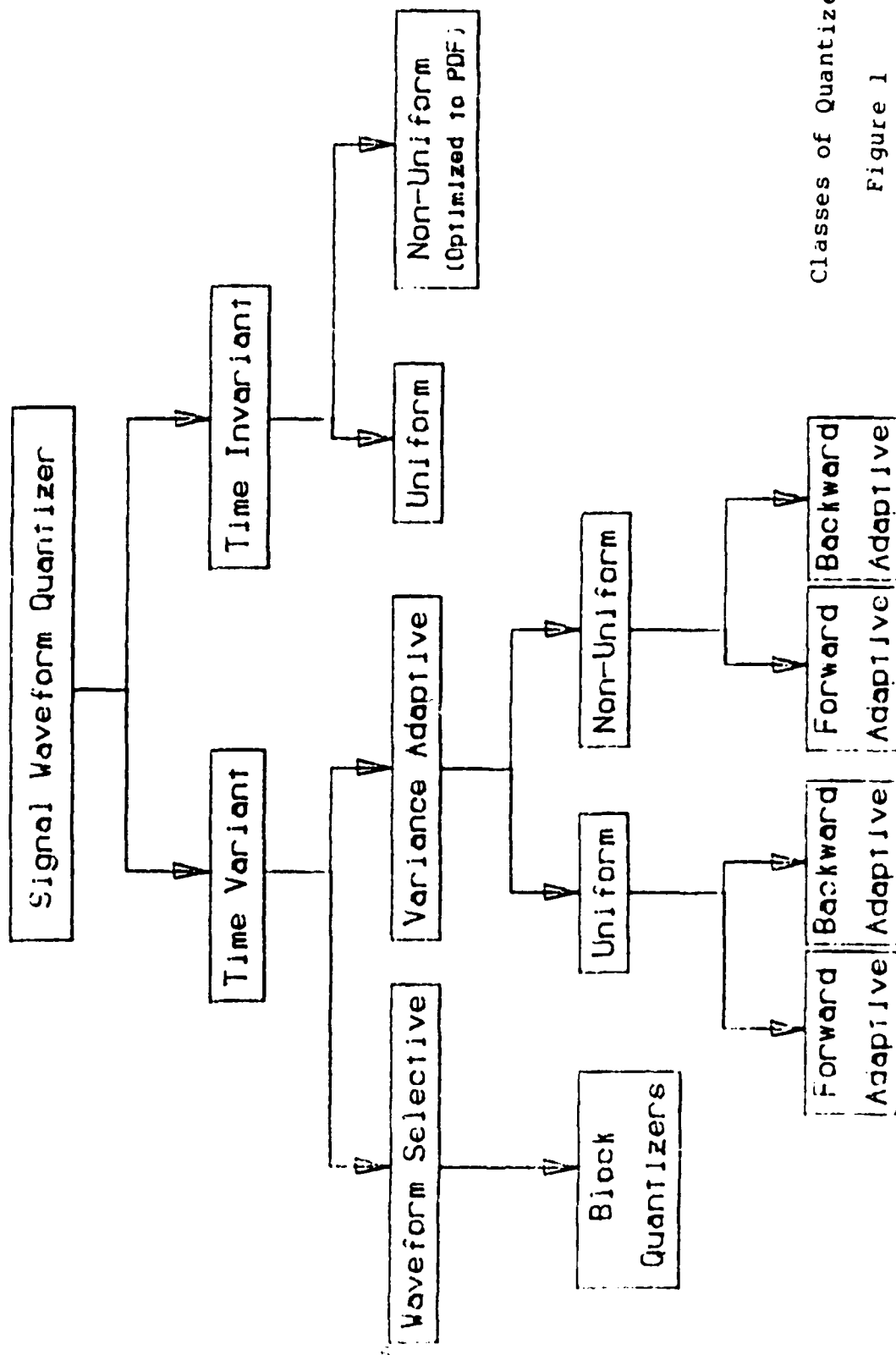
CHAPTER TWO

OVERVIEW OF QUANTIZER THEORY

In this chapter, the problems of selecting the appropriate methods of signal digitization needed to approximate a source signal waveform is addressed. In particular, an examination will be made of various types of signal quantization including theory, implementation, performance, and general applications of quantizers with the goal of guiding the selection of a quantization/coding scheme for image transmission.

Signal quantization may be defined as a mapping of samples from an analog source signal into a finite set of values (constrained alphabet) representing the samples in an attempt to approximate or identify the source. Methods of obtaining this mapping are of primary concern. It will be shown that the specific application, as well as performance and implementation criteria, will be of fundamental importance to the selection of appropriate methods.

Signal or waveform quantizers have been proposed in multitudinous variety. However each exhibit certain characteristics which can be classified. Some basic classifications of many quantizers are shown in figure 1. Each class of quantizers contains its own particular



Classes of Quantizers

Figure 1

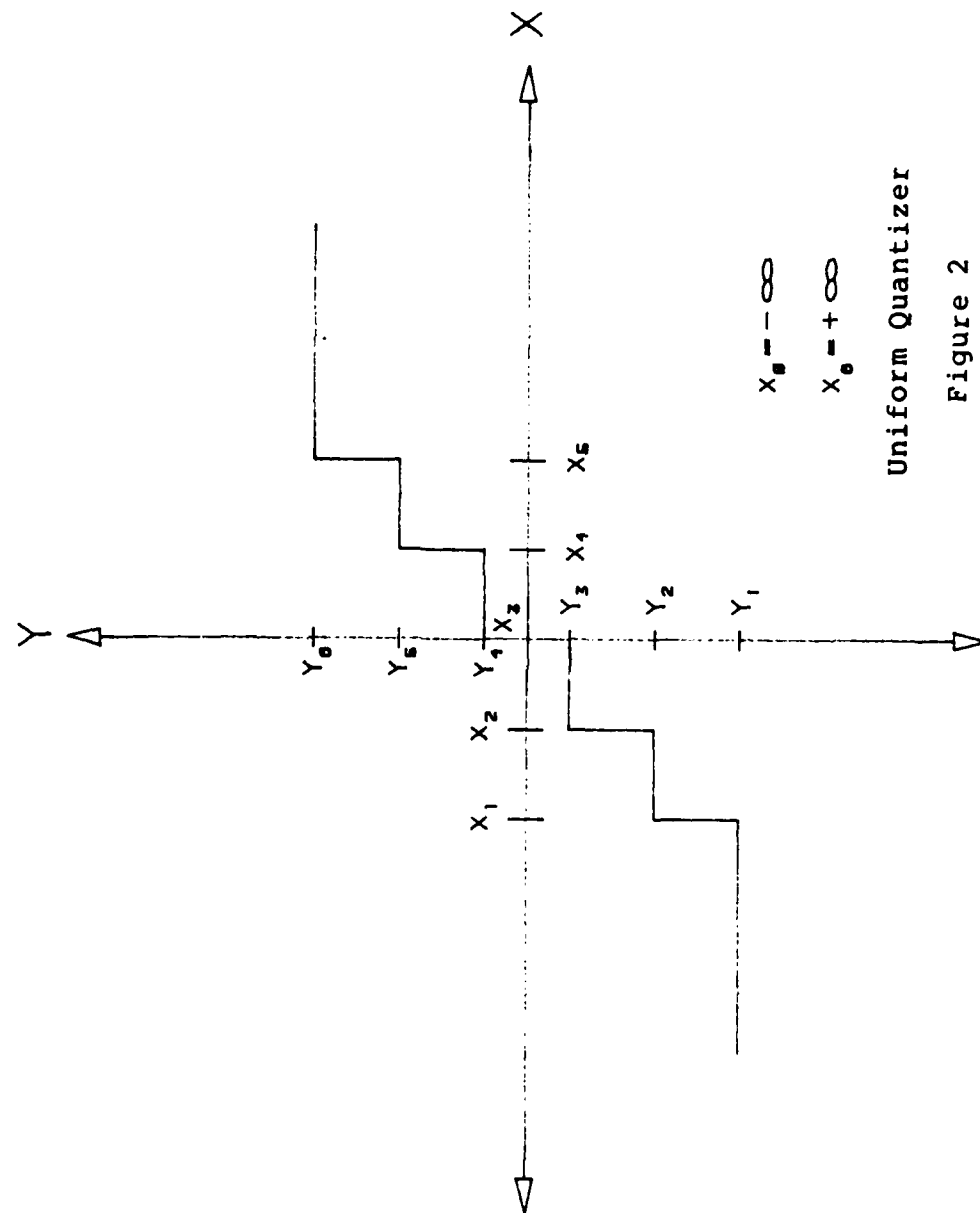
advantages and disadvantages due to theory and design tradeoffs.

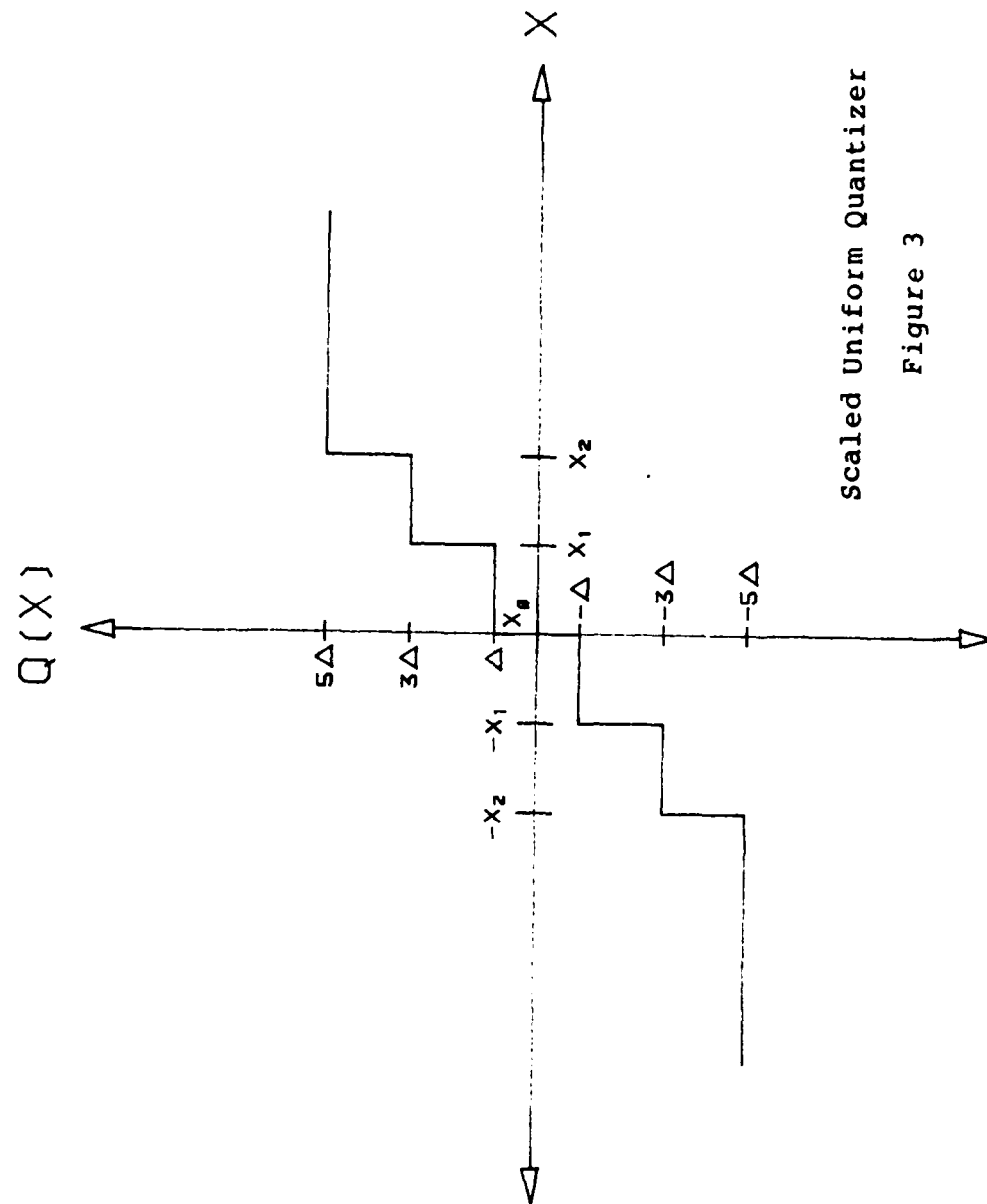
I. TIME-INVARIANT QUANTIZERS

A. UNIFORM QUANTIZATION

Gersho [11] describes memoryless N-point symmetrical quantizers by specifying a set of $N+1$ decision levels $\{x_k\}$, $k=0, \dots, N$, and a set of N output points $\{y_j\}$, $j=1, \dots, N$. When the value, x , of an input sample lies in the i th quantizing interval, namely $R_i = \{x_{i-1} < x < x_i\}$, the quantizer produces the output value y_i . Since y_i is used to approximate samples in R_i , it follows that $y_i \in R_i$. The outer levels are chosen so that $x_0 = -\infty$, $x_N = \infty$. See figure 2. The description of this class of quantizers certainly fits the general definition stated previously.

Probably the first widely-used quantizers were those of the type described by Gersho, particularly the one-dimensional time-invariant class. These provide easy implementation due to their simplistic design. A quantizer of this sort merely samples the signal to be quantized (normally at a fixed uniform sample rate) and immediately produces a representative value for each sample from a fixed





Scaled Uniform Quantizer

Figure 3

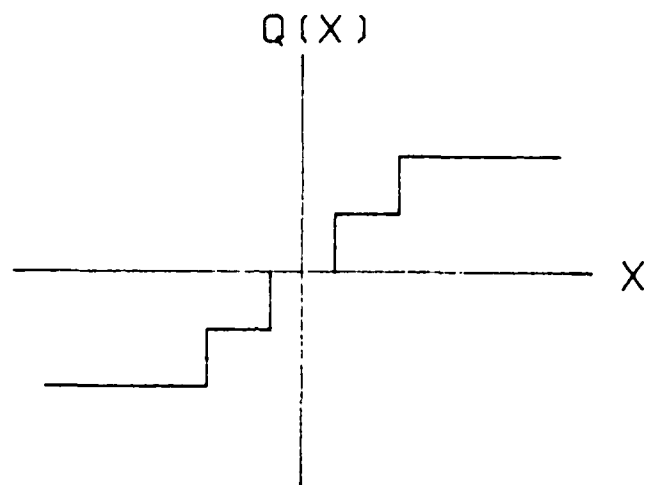
set of finite, uniformly spaced values symmetrically placed about the average input value. A typical input/output graph is illustrated in figure 3.

Two basic styles of design now surface: the mid-tread and the mid-riser. The mid-tread produces a zero output for signals at the mean input level. The mid-riser produces output values shifting from a negative value to a positive value at the mean input level. See figure 4. Note that the terms positive and negative could be with respect to a mean value instead of zero.

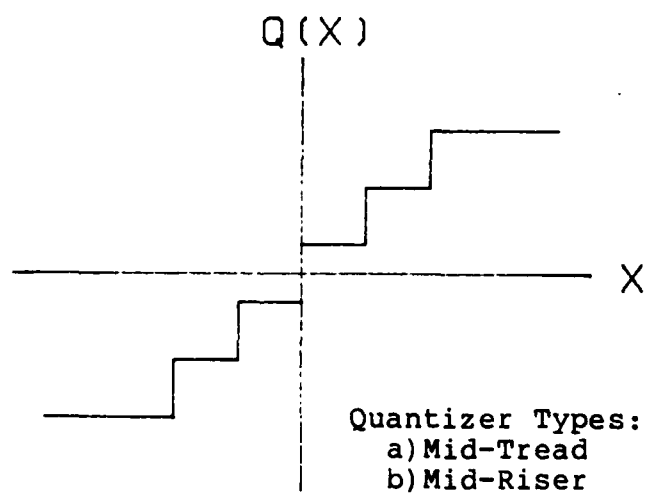
As is intuitively obvious, this type of quantizer would perform best for stationary uniformly distributed source signals. If the source statistics were to change with time, the quantizer could overload -- producing useless output. If we express the quantization distortion as $d(x) = Q(x) - x$ we find the minimum distortion for this system is produced when the input signal, x , is uniformly distributed. This is true regardless of whether it is of the mid-tread or mid-riser type.

B. NON-UNIFORM QUANTIZATION

Most signals of interest are seldom of the uniformly distributed class. Therefore, to accommodate arbitrary distributions, a design procedure must be developed.



4-a



Quantizer Types:
a) Mid-Tread
b) Mid-Riser

4-b

Figure 4

Assuming the signals to be stationary, the case of signals with known probability distributions is investigated. Applications where a specific probability density function is known to describe the source signal can use quantizers which are designed to take advantage of this special knowledge. Derivations of quantizers for these signals are presented in the classic works by Lloyd [24] and Max [25] and will be discussed extensively in a later chapter. These papers dealt with producing optimum memoryless signals with probability distributions known a priori.

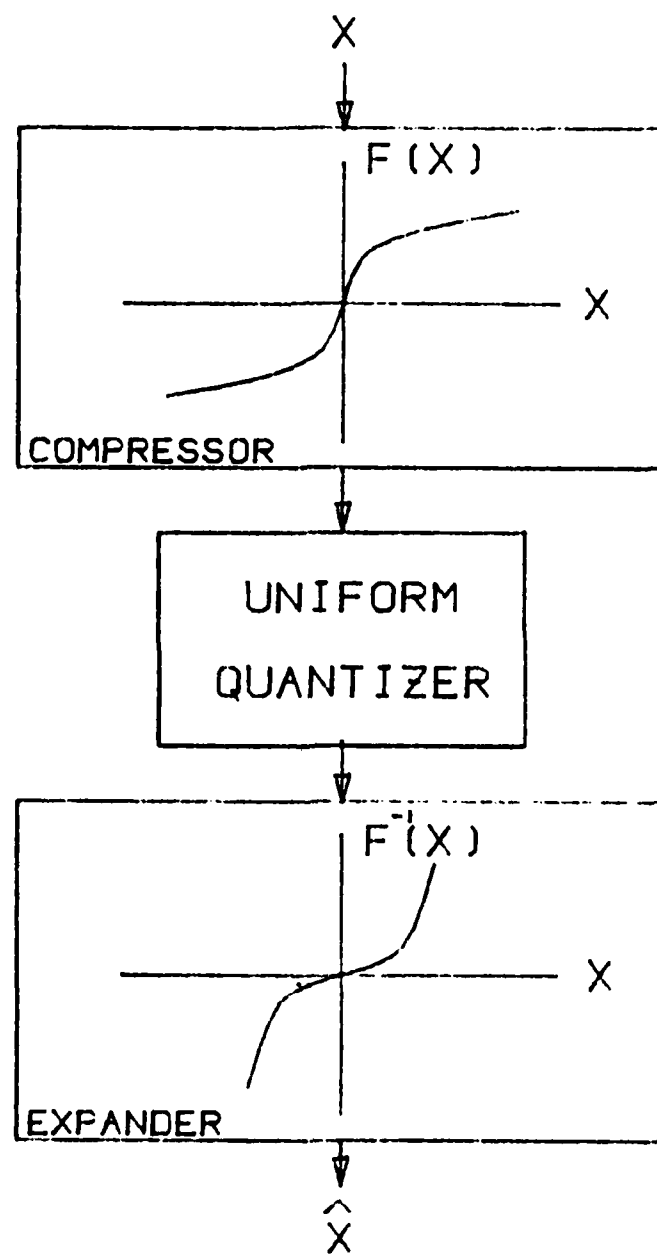
Quantizers which minimized distortion defined by the mean squared quantization error were specifically derived, yet the same general procedure could be used for distortion measures of other types. Roe [33], for example, derived quantizers for all distortions of the form: $E\{\text{error}^e\}$. Tabulated results for the decision levels and output points were produced for a standard normal distribution from the iterative procedure derived [24] [25]. Others have since formed tables for other common distributions, such as the Laplacian and Rayleigh, useful in speech, optical holography, and image processing applications [22] [23].

C. COMPANDING

Another method for digitizing signals of this nature has been suggested by Bennett [2]. The non-uniform quantizer is modeled as a zero-memory nonlinearity function, $F(x)$, followed by a uniform quantizer, in turn followed by the inverse of the function, $F^{-1}(x)$. The procedure is called "companding". A diagram appears in figure 5. The first nonlinear function, $F(x)$, compresses the input signal by spreading out the low amplitude portions and shrinking those of high amplitude. Since the low amplitudes have a higher probability, this allows a larger proportion of decision levels of the uniform quantizer to be devoted to higher probability of occurrence regions, and fewer to those of lower probability -- increasing the accuracy of the representation. The inversion at the end reverses the process, producing the estimate of the signal. This compression and expansion of the signal, hence the name "companding", combine to yield results comparable to those of the non-uniform quantizer.

Holzwarth and Smith [11] have made a study of speech transmission systems with widely varying power levels using companding. They determined a compressor function of:

$$F(x) = V \frac{\log(1-\mu x(v))}{\log(1+\mu)}$$



"Companding"

Figure 5

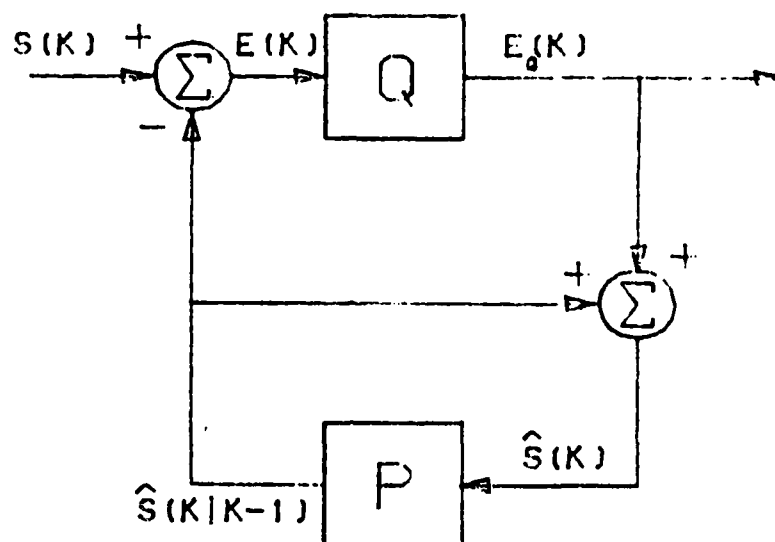
Since known as a " μ -law curve".

II. TIME-VARYING QUANTIZERS

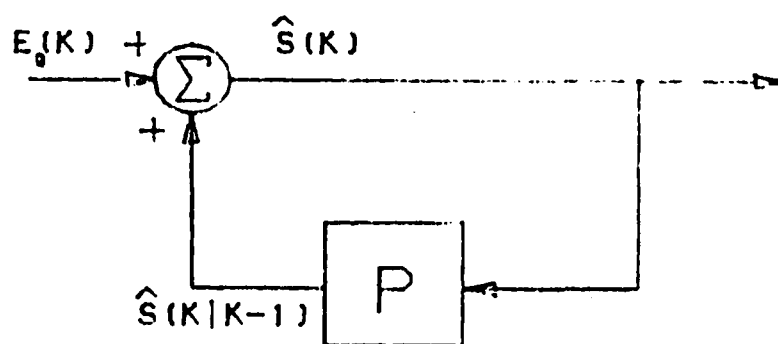
A. ADAPTIVE QUANTIZERS

Turning now to the non-stationary sources, it is only natural that research continue from achievements made thus far. Certain applications such as speech digitization and image coding systems must deal with this type of source. Quantizers which can adapt to time-varying source statistics and still yield satisfactory results are vital to such data compression schemes. Whichever scheme one uses, such as APC, DPCM, or PDPDCM, the quantizer will determine the overall system performance. One common performance measure is the signal-to-quantization noise ratio.

Most of the recent work in speech digitization and image coding systems concentrate on the use of designs related to the DPCM (Differential-Pulse-Coded-Modulation) configuration. Gibson [15] gives a good tutorial on such designs. The aspects of fixed and adaptive predictors utilized in such systems will not be presented here, but it should be noted that they too will play an important role in system performance and efficiency by reducing signal redundancy such as speaker pitch in speech systems. The



(A) DPCM TRANSMITTER LOOP



(B) DPCM RECEIVER LOOP

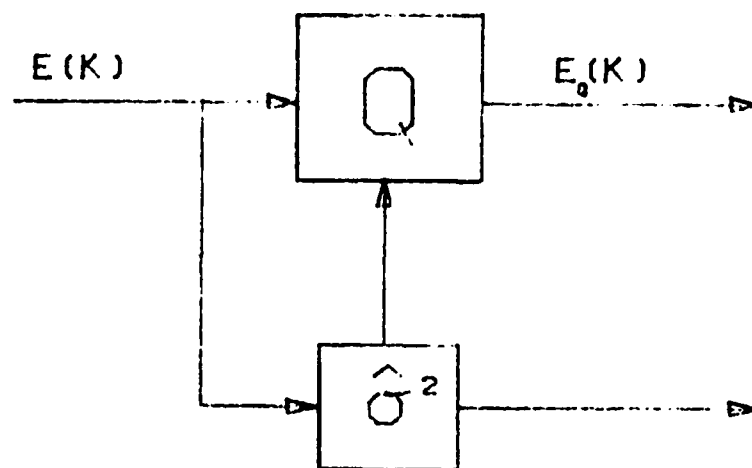
DPCM a) Transmitter Loop
 b) Receiver Loop

Figure 6

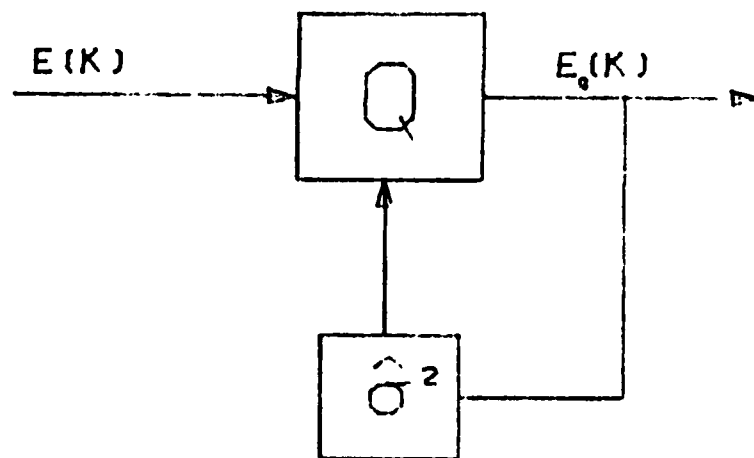
general DPCM differential encoder and decoder block diagrams are illustrated in figure 6. Systems using these schemes without any adaptive consideration are optimal only for stationary processes.

For the quantizer to adapt to the signal's statistical variations, it must acquire some form of estimate of these statistical variations, usually the signal variance, upon which it is to base its adaptation. Two common classifications of adaptive quantizers are provided by separating quantizers as to whether they are forward adaptive or backward adaptive. Forward adaptive quantizers extract the information for their estimate from the quantizer input and must transmit some additional information regarding the subsequent step size variation to the receiver. Backward adaptive quantizers utilize the quantized signal alone, eliminating the need for additional transmission. The backward adaptive quantizer must however, base its estimate on cruder information due the quantization noise. See figure 7.

The basic idea is to modify the quantizer input and output step sizes in an effort to match the changing signal using a signal variance estimate. Goodman and Gersho [17] describe well the adaptation. Again the quantizer of $2N$ output levels is characterized as in figure 8. It contains a set of $N+1$ decision levels $\{\Delta(k)\xi_i\}$, $0 < i < N$, and a set of N



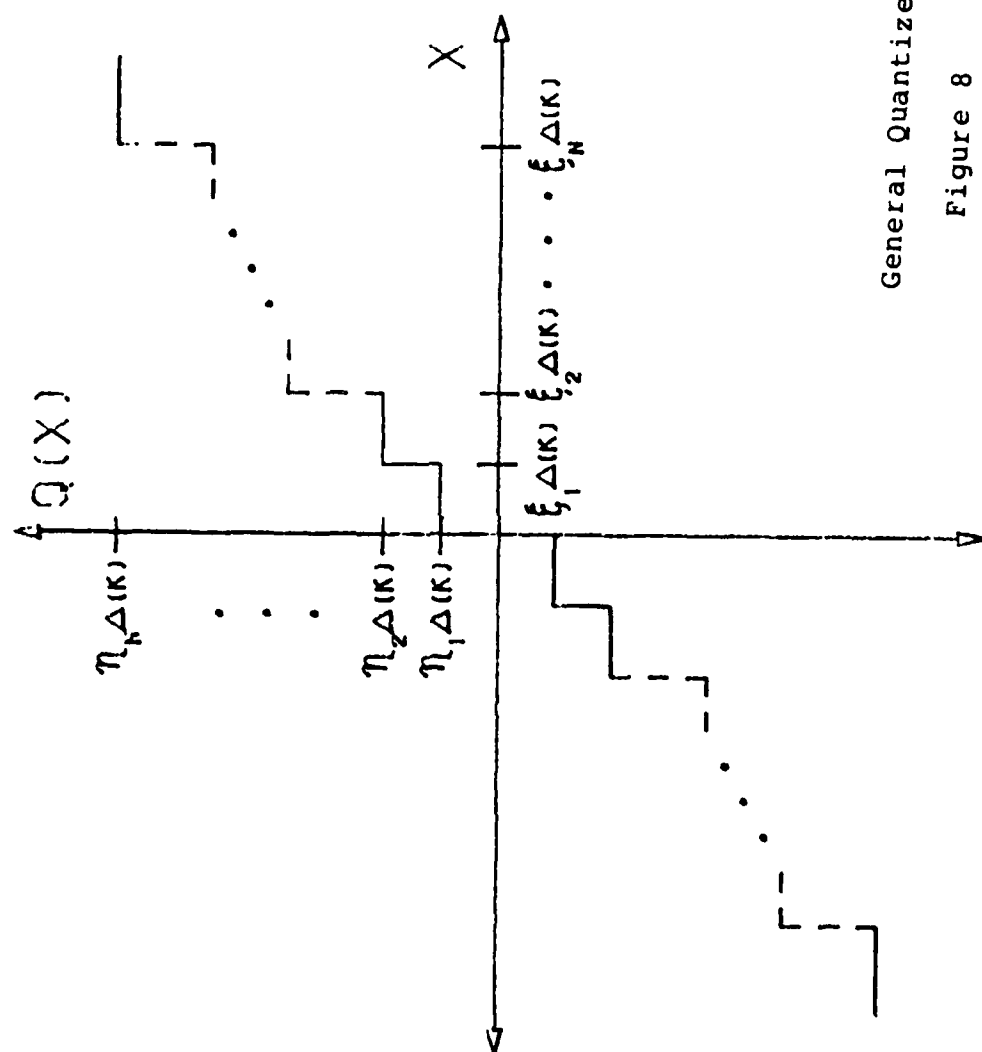
(A) FORWARD ADAPTIVE QUANTIZER



(B) BACKWARD ADAPTIVE QUANTIZER

Adaptive Quantizer: a) Forward Adaptive
b) Backward Adaptive

Figure 7



General Quantizer

Figure 8

quantization levels, $\{\Delta(k)\eta_i\}$, $1 \leq i \leq N$. For an input $z(k)$, the quantizer is given by:

$$Q(z(k)) = \Delta(k)\eta_i \quad z > 0; \quad 0 < \Delta(k)\xi_{i-1} < z(k) < \Delta(k)\xi_i \quad (1a)$$

and by symmetry:

$$Q(z(k)) = -Q(-z(k)) \quad z < 0. \quad (1b)$$

The parameters $\{\xi_i\}$ and $\{\eta_i\}$ are forced to approximate the shape of the desired probability density determined through the distortion minimization analysis such as that by Lloyd [24] and Max [25] for stationary sources. The scaling factor, or step size $\Delta(k) > 0$, which determines the dynamic range, will be made to vary with time. Several good algorithms for determining step size have been formulated, such as these three given by Stroh [15]:

$$\Delta(k) = a \sqrt{1/k \sum_{i=1}^k e^2(k-i)}, \quad (2)$$

$$\Delta(k) = a_1 \left[(1-a_2) \sum_{i=1}^k a_2^{i-1} e^2(k-i) \right]^{1/2}, \quad (3)$$

$$\Delta(k) = \left(\frac{1-a_1}{a_2} \right) \cdot \sum_{i=1}^k a_2^{i-1} \cdot e(k-i), \quad (4)$$

for both forward and backward adaptive quantizers to be used in a DPCM scheme. Noll [15] proposed two:

$$\Delta(k) = \alpha \max\{|s(k-i) - s(k-i-1)|\}, \quad (5)$$

for first order predictors and

$$\Delta(k) = \alpha \left\{ \sum_{i=1}^k [s(i) - \sum_{j=1}^N a_j s(i-j)]^2 \right\}^{1/2} \quad (6)$$

where $\{a_j, j=1, \dots, N\}$ are N predictor coefficients for an N th order predictor. Jayant and Cummiskey [15] obtained the popular "one-word memory" quantizer with step size:

$$\Delta(k+1) = M(|Q(k)|) \Delta(k) \quad (7)$$

which is a backward adaptive quantizer. $M(\cdot)$ is a time-invariant multiplier function dependent upon the quantized signal as defined in (1) with $z(k) = e(k)$; $e(k)$ some residual error signal such as in DPCM.

It has been found that an adaptive quantizer based on a step size as in (7) performs well for ideal channels, but is quite susceptible to transmission errors. Gibson and Wilkenson [15] have shown that the difficulty can be mitigated with the algorithm modified to:

$$\Delta(k+1) = M(|Q(k)|) \cdot \Delta^{\beta}(k) \quad (8)$$

where $0 < \beta < 1$ will provide a degree of robustness. Gibson [15] discusses a closely related backward adaptive quantizer proposed by Cohn and Melsa [9] [10] where the step size adjusts in the same fashion as the adaptive Jayant quantizer

except for two rarely used outer levels. The purpose of which is to allow rapid expansion when a pitch pulse occurs in a speech encoding system. It is thus termed a pitch-compensating quantizer (PCQ), and has seen principal applications with fixed and backward adaptive predictors.

B. CODING

Once the signal is quantized, it is usually encoded in some fashion for transmission. The most common form of coding for an N-level quantizer is binary coding. The idea being to assign a code word of length $\log_2 N$ for each output level. This is based on the assumptions that the symbols are independent and all quantization levels are equally likely. The assumption of source symbol independence is generally untrue for most applications. Huang and Schultheiss [20] offer a technique to remedy this problem and will be discussed momentarily.

The assumption of equally likely quantization levels is also untrue for applications such as speech. Researchers have taken advantage of this fact by designing variable-length codes for the output levels. Using a priori knowledge, one can assign code words of short length for highly probable quantization levels and code words of longer length for those levels with lower probability. Variable

length coding is also known as "entropy coding", since the average code word length can approach the entropy of the symbols transmitted when the symbols are indeed independent. Hence, the entropy given as:

$$H = - \sum_{i=1}^N p_i \log_2 p_i \quad (9)$$

becomes a lower bound for the average code word length. Berger [4] defines minimum entropy quantizers in a rate distortion sense by defining a rate:

$$R = - \sum_{i=1}^N p_i \log_2 p_i \quad (10)$$

and distortion:

$$D = E|Y-X|^r = \sum_{i=1}^N \int_{a_{i-1}}^{a_i} |x - \gamma_i|^r dF(x) \quad (11)$$

where $\{a_i\}$ is the set of quantization thresholds and $\{\gamma_i\}$ the set of reconstruction levels. The optimum entropy quantizer is therefore one which minimizes the distortion D for a fixed rate R .

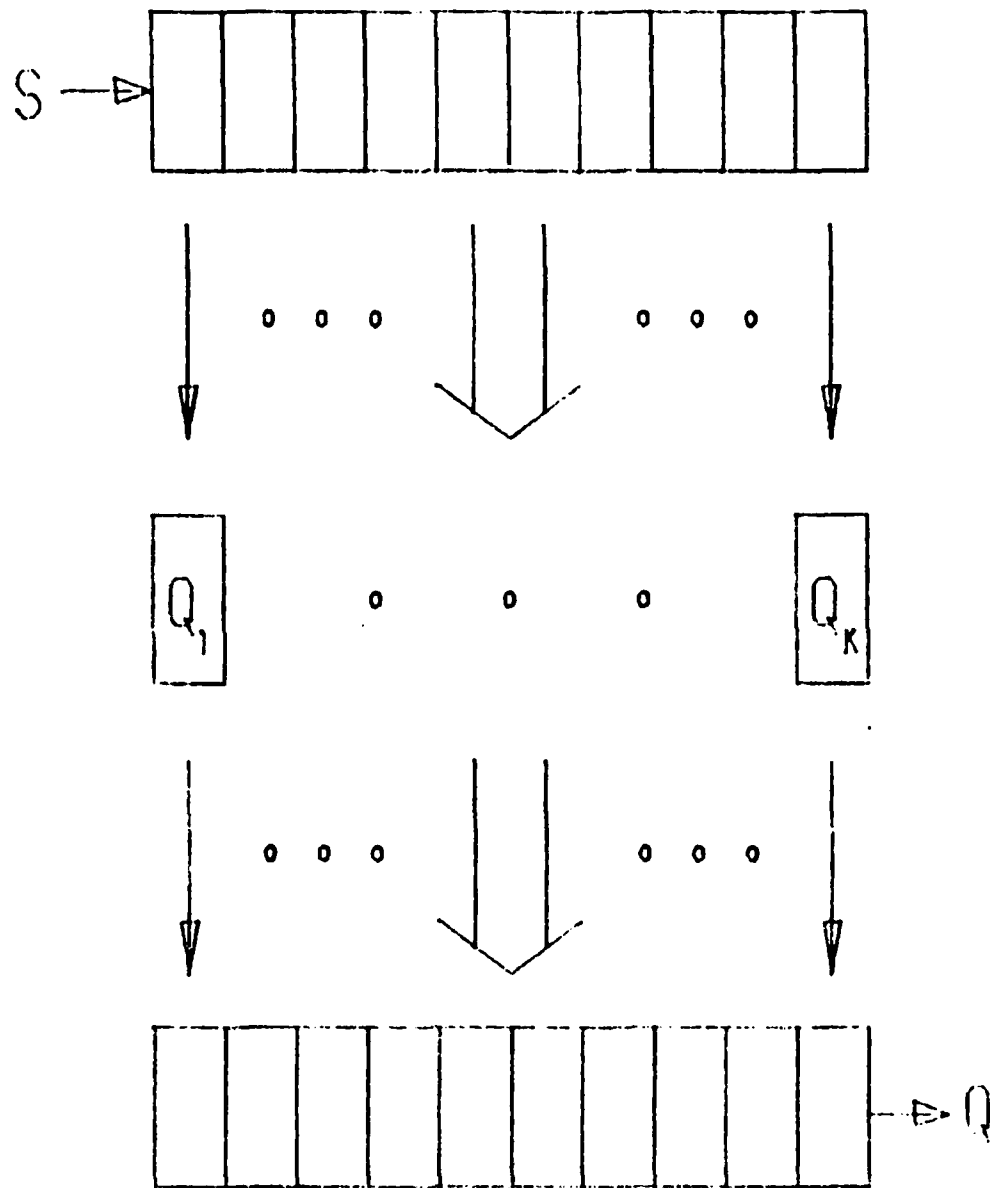
Berger [4] also discusses permutation coding as developed by Slepian [34] and introduces a theorem showing variable-length coding of quantizer output and permutation coding are equivalent in the sense that their optimum R versus D performances are identical.

C. VECTOR QUANTIZERS

All that has been discussed previously is now extended to the next level of sophistication. This new level is one which will deal with quantizing multiple or "blocks" of samples called quantization with memory. Block quantization is exactly that, quantizing the samples in blocks. One can think of the quantizers discussed previously as one-dimensional quantizers, and block quantizers operating on blocks of K samples as K -dimensional. The procedure uses blocks of quantizers for transmission as in figure 9.

Huang and Schultheiss [20] incorporate this method in a system for the transmission of correlated gaussian random variables. They use a linear transformation (P) to first convert K dependent random variables into K independent random variables. These are quantized one at a time and the output of each quantizer is transmitted by a binary code. Lastly, a second $K \times K$ linear transformation (R) constructs from the quantized values the best estimate (in the mean-squared-error sense) of the original values. They also show that the best transformation R would be the inverse of the first transformation P^{-1} . The best transformation P is the transpose of the orthogonal matrix which diagonalizes the moment matrix of the original (correlated) random variables. A diagram of this process is illustrated in figure 10.

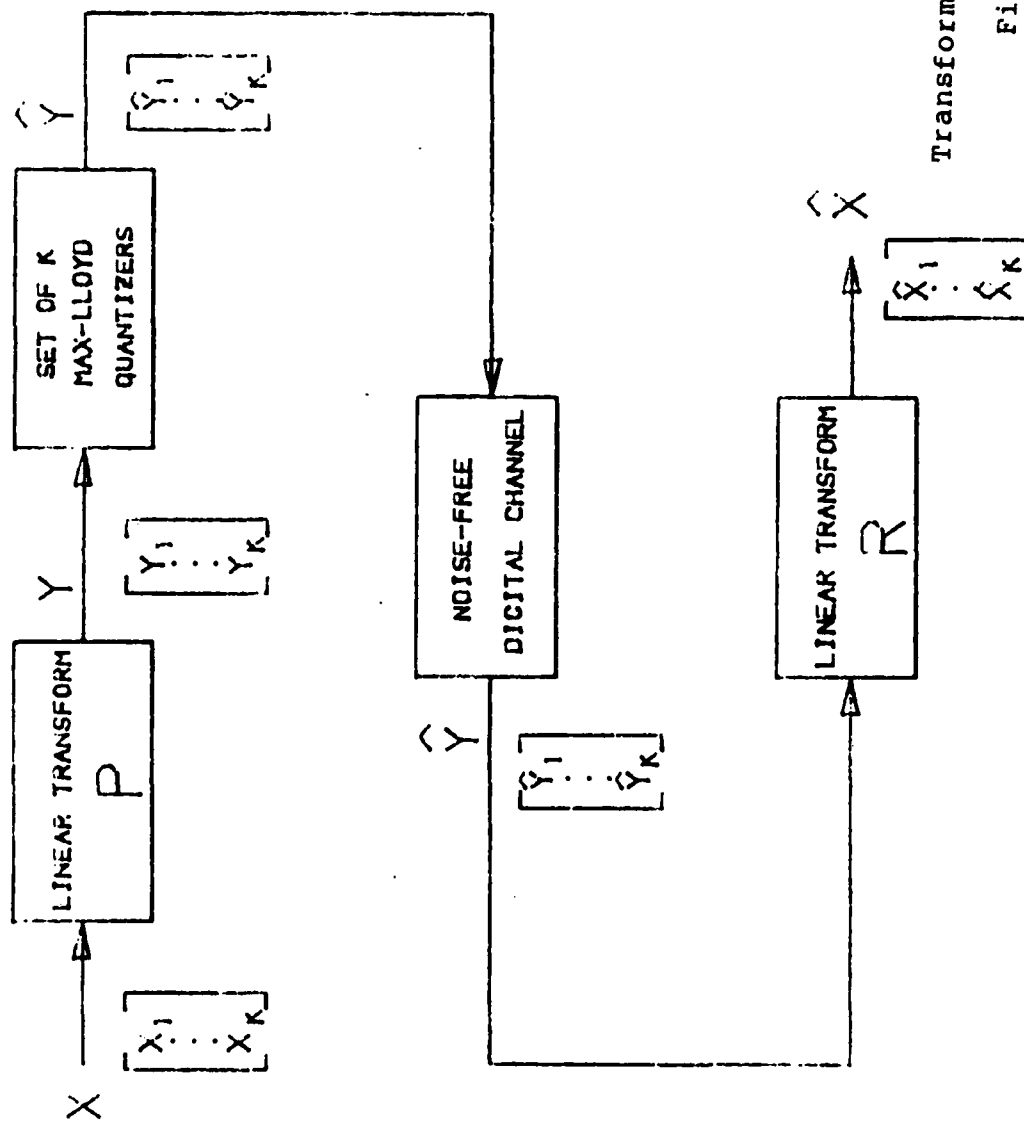
K SAMPLES



K QUANTIZED VALUES

Block Quantizer

Figure 9



Transform Coding System
Figure 10

Although the terms block and vector quantization are often used interchangeably, when one sees the term vector quantization it is usually in reference to a method more subtle than the brute force method of K quantizers. Gersho [13] states that block or vector quantization is intrinsically superior to predictive coding, transform coding, and other suboptimal procedures since it achieves optimal rate distortion performance subject only to memory (block) length of the observable segment being encoded. One can think of vector quantization as a method of representing a sequence of K samples of the signal as a vector in K -dimensional space. This space is partitioned into a finite number of regions, the geometry of which is determined in a manner such that certain optimality criteria are achieved. Each region of space is then assigned a specific code word. As the region of space for each sample vector is identified, its representative code word is selected and after transmission can be decoded via a code lookup table or "code book" into an output vector representing the original input vector. Thus vector quantization is a surjective mapping of K -dimensional input vectors to corresponding representative vectors,

$$Q : R \xrightarrow{k} Y, \quad Y \subset R^k \quad (12)$$

D. RANDOM QUANTIZERS AND CODE BOOK DESIGN

Gersho [13] analyzes the approach to vector quantization by Linde, Buzo, and Gray [23] with random quantizers. Since the only effective method for the design of multidimensional quantizers is the use of a clustering algorithm, their design algorithm uses a training set of random vectors generated from the source. The training set is the collection $\{x_1, \dots, x_M\}$ of M independent observations of the continuously distributed random variable X with a specified joint density function where $M \gg N$ for an N level or N regional quantizer. The set of output vectors $\{y_1, \dots, y_N\}$ is determined by:

$$y_j = \frac{\sum_{i=1}^M x_i S_{ji}(x_i)}{\sum_{i=1}^M S_{ji}(x_i)} \quad (13)$$

and

$$\|Q(x_i) - x_i\| \leq \|y_j - x_i\| \quad (14)$$

for all $i=1, 2, \dots, M$ and $j=1, 2, \dots, N$.

The code book resulting from such a cluster design may contain a high degree of complexity in coder implementation. This arises from the fact that an exhaustive search of the entire code book must be made for the nearest output vector to the input vector since the output vectors have no natural structure. However, at the price of suboptimality and increased code book storage requirements, one can induce certain structures upon the code book to greatly reduce the search time.

Buzo, Gray, Gray, Markel [8] use a binary tree-structured code book in a speech coding problem. Juang, Wong, Gray [22] show a more general M-ary tree structured code book design offers better tradeoff between performance and processing time but is yet inferior to a "full-search" code book.

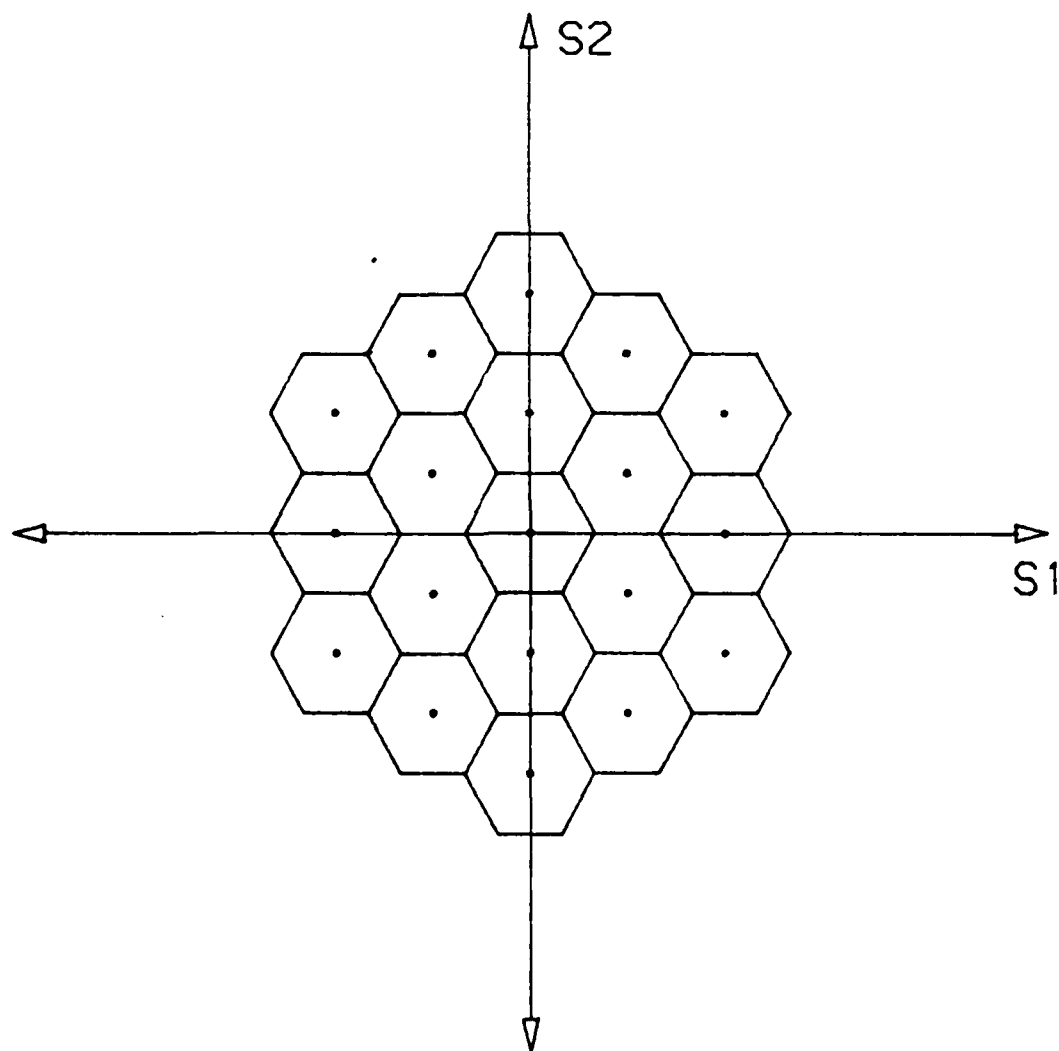
E. LATTICE QUANTIZERS

Another structured code book design is based upon lattice theory. Gersho [13] thoroughly describes lattice quantizers. The quantizer uses a set of output points which lie in a bounded region of a lattice. A K-dimensional lattice being defined by any nonsingular KxK matrix (U) so that for a K-dimensional column vector (m), the lattice (Λ) is given by the set of all vectors of the form:

$$\Lambda = U m \quad (15)$$

Notice that the columns of U are lattice points and form a basis. So all other points may be found by taking linear combinations of the basis vectors with integer-valued coefficients. An example of a two-dimensional hexagonal lattice quantizer is depicted in figure 11. The hexagon shapes define the partition regions of the decision space, and the points correspond to the representative values for each partition set. Note that the points are in fact the centroid of each hexagon.

Since the lattice points form a regularly spaced array of points in K-dimensional space, a lattice quantizer is a uniform quantizer. Gersho [19] uses this fact in designing multidimensional companders, much as those of the one-dimensional class are designed.

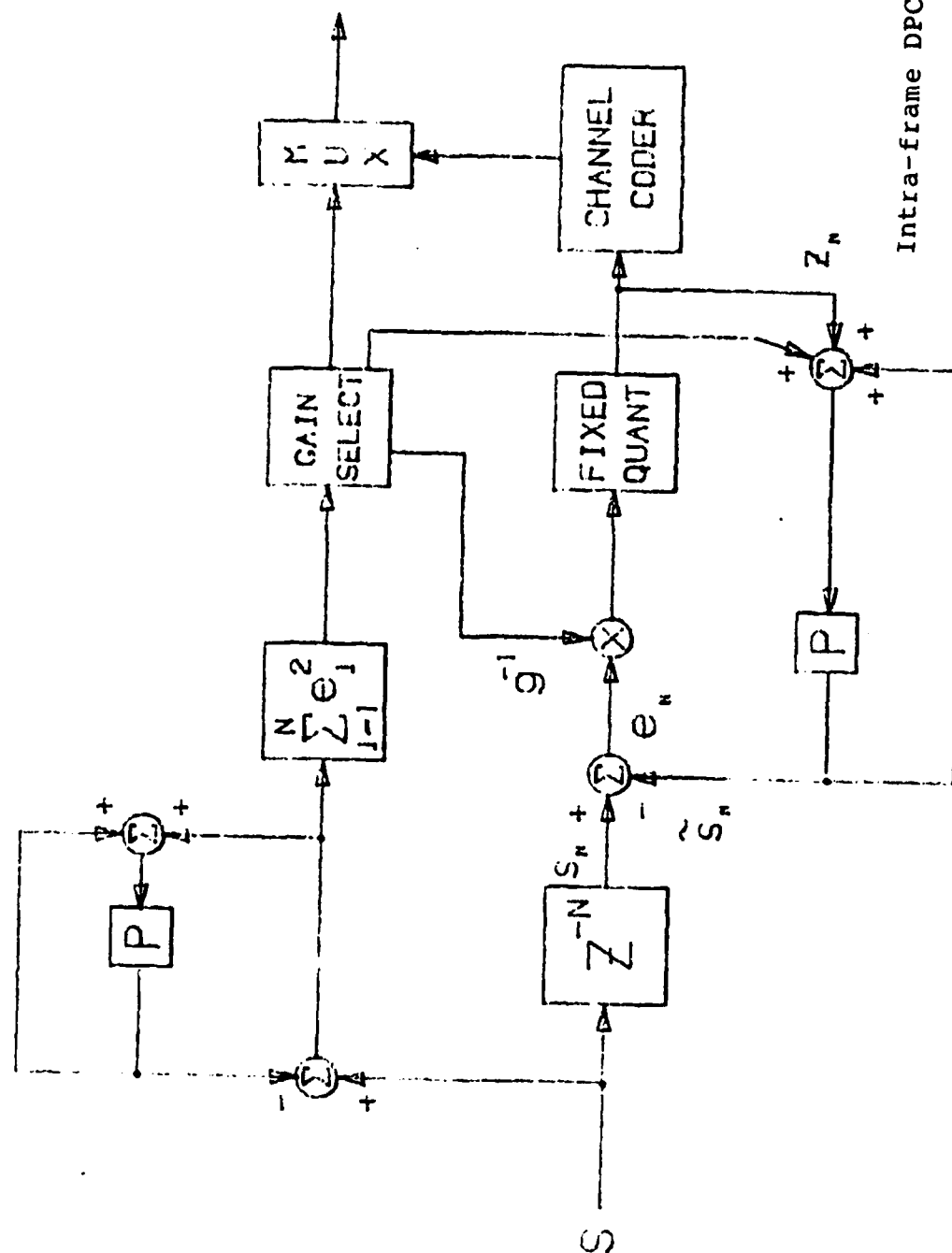


Lattice Quantizer

Figure 11

Intra-frame DPCM using adaptive quantization has recently been studied by Zetterberg [38] for two types of adaptive quantizers discussed earlier -- the Jayant quantizer and a forward adaptive quantizer using a variance estimator. Comparisons were made and tabulated by judging picture quality via SNR and also subjective examination.

They observed that nonmoving areas should use a fine quantizer to limit granular noise to low levels whereas moving areas need a large dynamic range or coarser quantizer to avoid overload by large prediction values. Entropy coding was then used in order to meet a 1 bit/pel requirement of European standard transmission rate. The basic block diagram is figure 12.



Intra-frame DPCM System

Figure 12

CHAPTER THREE

THE ASPC SYSTEM

A number of well known techniques in communications theory exist for compressing data from a source signal. Two methods which have received much attention are the Differential Pulse Coded Modulation (DPCM) method and the transformation method. Both exhibit the desired data compression performance, yet their operation is quite different.

I. DPCM SYSTEMS

DPCM systems operate under the assumption that a degree of correlation exists among a stream of data that is to be transmitted. This assumption is the basis for a scheme whereby a prediction of each datum can be made based on the previous data. In other words, a prediction at time k , is made on the next piece of data to be transmitted, at time $k+1$, based upon a linear combination of the present and previous data. So that,

$$S(k+1|k, k-1, \dots, k-n) = \sum_{i=0}^n a_i S(k-i) \quad (16)$$

where $S(k)$ represents the data stream to be transmitted,

$S(k+1 | k, k-1, \dots, k-n)$ the predicted value for time $k+1$ based on the previous values back through time $k-n$, and $\{a_i\}$ is a set on $n+1$ coefficients.

This prediction is then subtracted from the actual value of S at time $k+1$, $S(k+1)$, yielding an error or "residual" from the original sequence, called $E(k+1)$:

$$E(k+1) = S(k+1) - S(k+1 | k, k-1, \dots, k-n) \quad (17)$$

This residual is then quantized and transmitted. See figure 13-A. If the predictor and quantizer are adequately designed to the signals, then the desired data compression is achieved.

The receiver stage is assumed to have the same predictor as the transmitter, but since it will only receive a quantized estimate of the true residual, the transmitter must be restricted to operate with the quantized residuals in generating each update in order to achieve parity between the two. Therefore the quantized residual is added to the prediction to obtain the source estimate to be used in later predictions. So in fact,

$$S(k+1 | k, k-1, \dots, k-n) = \sum_{i=0}^n a_i S(k-i) \quad (18)$$

where

$$S(k) = S(k|k-1, \dots, k-n-1) + E_q(k) \quad (19)$$

and the quantized residual $E_q(k)$ is

$$E_q(k) = S(k) - S(k|k-1, \dots, k-n-1) + Q_n(k) \quad (20)$$

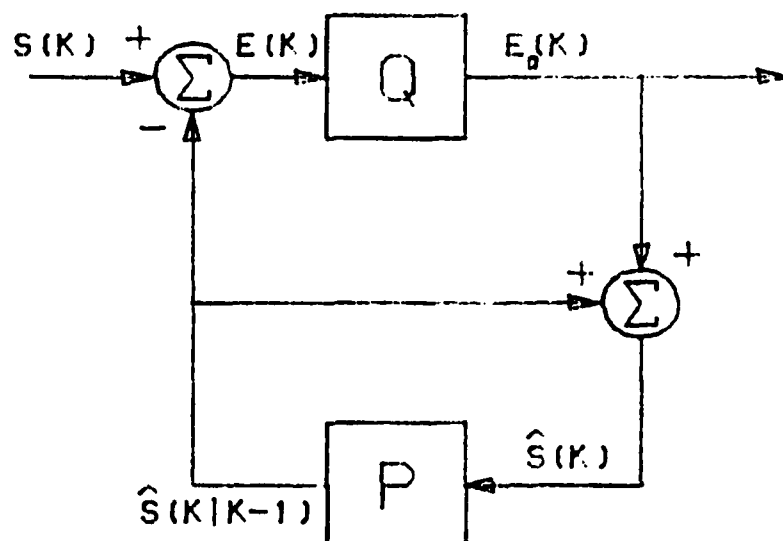
Thus the observed or estimated sequence is:

$$\begin{aligned} S(k) &= S(k|k-1, \dots, k-n-1) + \\ &\quad S(k) - S(k|k-1, \dots, k-n-1) + Q_n(k) \\ &= S(k) + Q_n(k) \end{aligned} \quad (21)$$

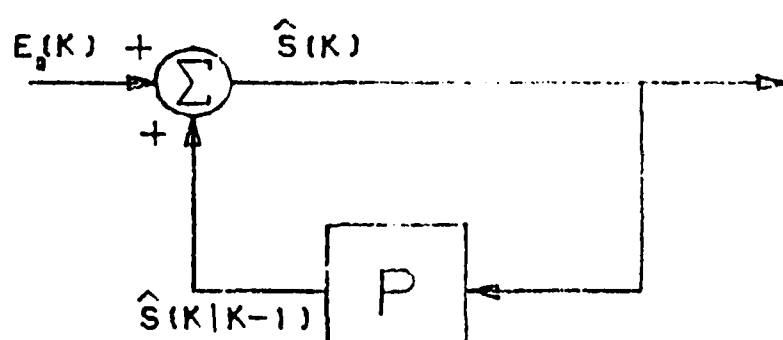
is the actual sequence $S(k)$ corrupted by quantization noise, $Q_n(k)$.

At the receiver, this process is mirrored by reconstructing the estimated sequence of the source by adding the prediction to the received quantized residual. See figure 13-B. Both predictors of the transmitter and receiver incorporate an n -cell memory register for the previous n values.

So if one neglects the effects of channel errors in the transmission and any external noise injected into the transmitter or receiver, then the quantization noise Q_n



(A) DPCM TRANSMITTER LOOP



(B) DPCM RECEIVER LOOP

DPCM a) Transmitter Loop
b) Receiver Loop

Figure 13

proves to be the detrimonious aspect of the system for a particular predictor scheme. Thus for a fixed prediction scheme, the quantizer yields the limiting influence to the system's performance. This is why so much energy has been devoted toward the study of quantizers for this communications configuration.

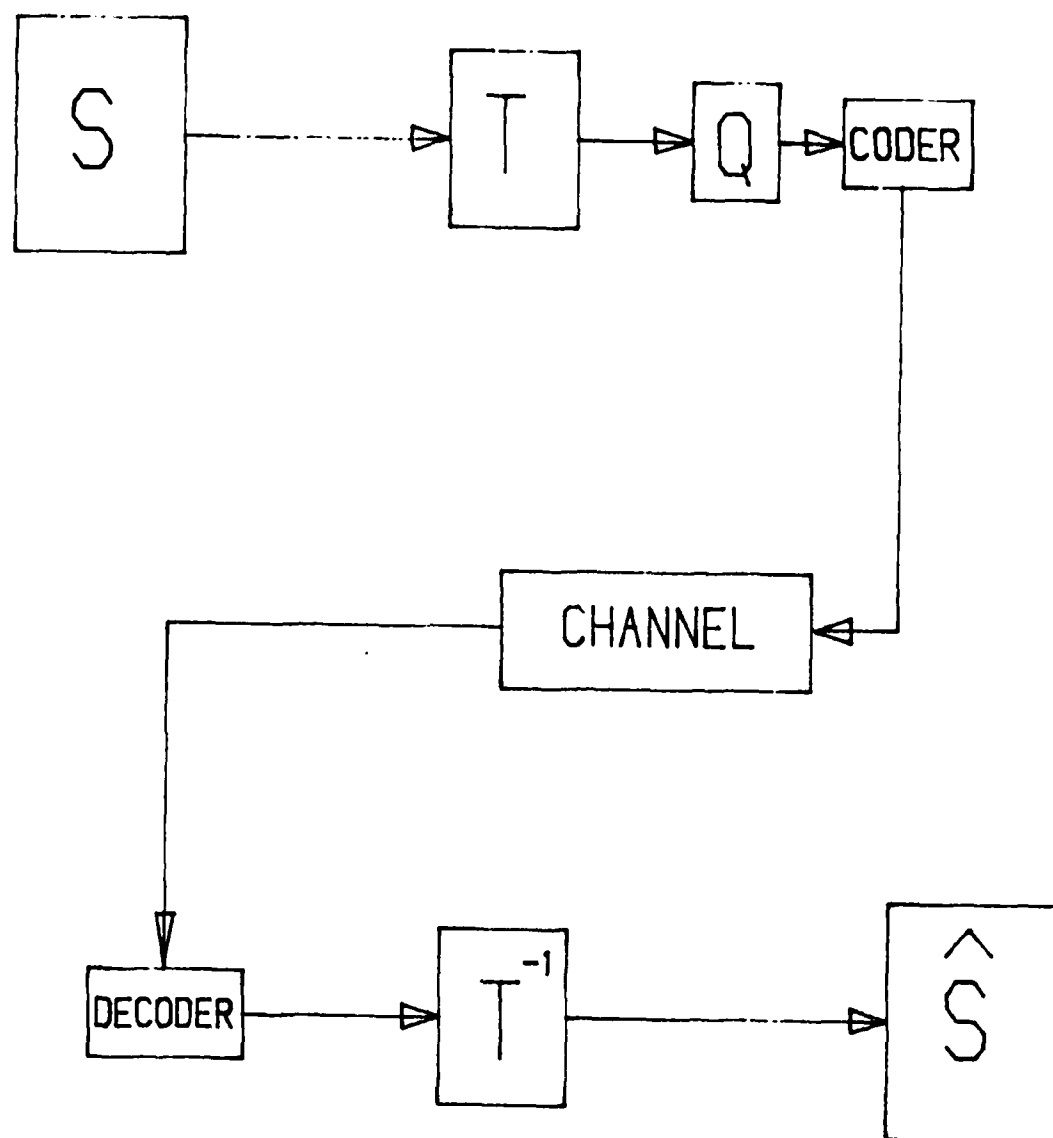
II. TRANSFORMATION SYSTEMS

According to Pratt [22], the notion of coding and transmitting the Fourier transform of a monochromatic image in place of the original image, was introduced by Andrews and Pratt in 1968. Images of this type tend to have a high degree of correlation among neighboring pixels. This correlation causes the energy distribution in a two-dimensional transform of a monochromatic image to be clustered into small portions of samples. It was discovered that to achieve a reduction in data rate, and thus bandwidth requirements, only the high order transform coefficients need to be quantized accurately to maintain a satisfactory reconstruction of the image.

The normal procedure in two-dimensional image coding is to apply a unitary transformation to an image which has been divided up into equal size square blocks. The highest energy transform coefficients in each block are finely

quantized while the lower energy coefficients are grossly quantized. These quantized values are then encoded and transmitted block-by-block to the receiver which performs the appropriate decoding and inverse transformation to reconstruct each original block, and hence the original picture. See figure 14.

Since this concept was first introduced, many have studied the effects of various unitary transformations and block sizes on the image reconstruction quality and bandwidth reductions. Some of the transformations examined include the Karhunen-Loeve, Hadamard, Haar, Slant, and Cosine transforms. The Karhunen-Loeve transformation, consisting of a matrix of eigenvectors of the data to be transformed, provides a total decorrelation of the block to give the lowest distortion of any of the transforms. But the problem with this transformation is the requirement of an exact statistical knowledge of the image block to be transformed, which is normally unavailable. Even if it is available, the computation of the transform is slow and complex. Sub-optimal approximations of the Karhunen-Loeve transform have been made by assuming the original image to be of a known specific statistical nature, such as a first-order Markov process. If this type of assumption is made, then the computation of the eigenvector transformation matrix becomes fast and simple. Unfortunately this is much



2-Dim Transform Coding

Figure 14

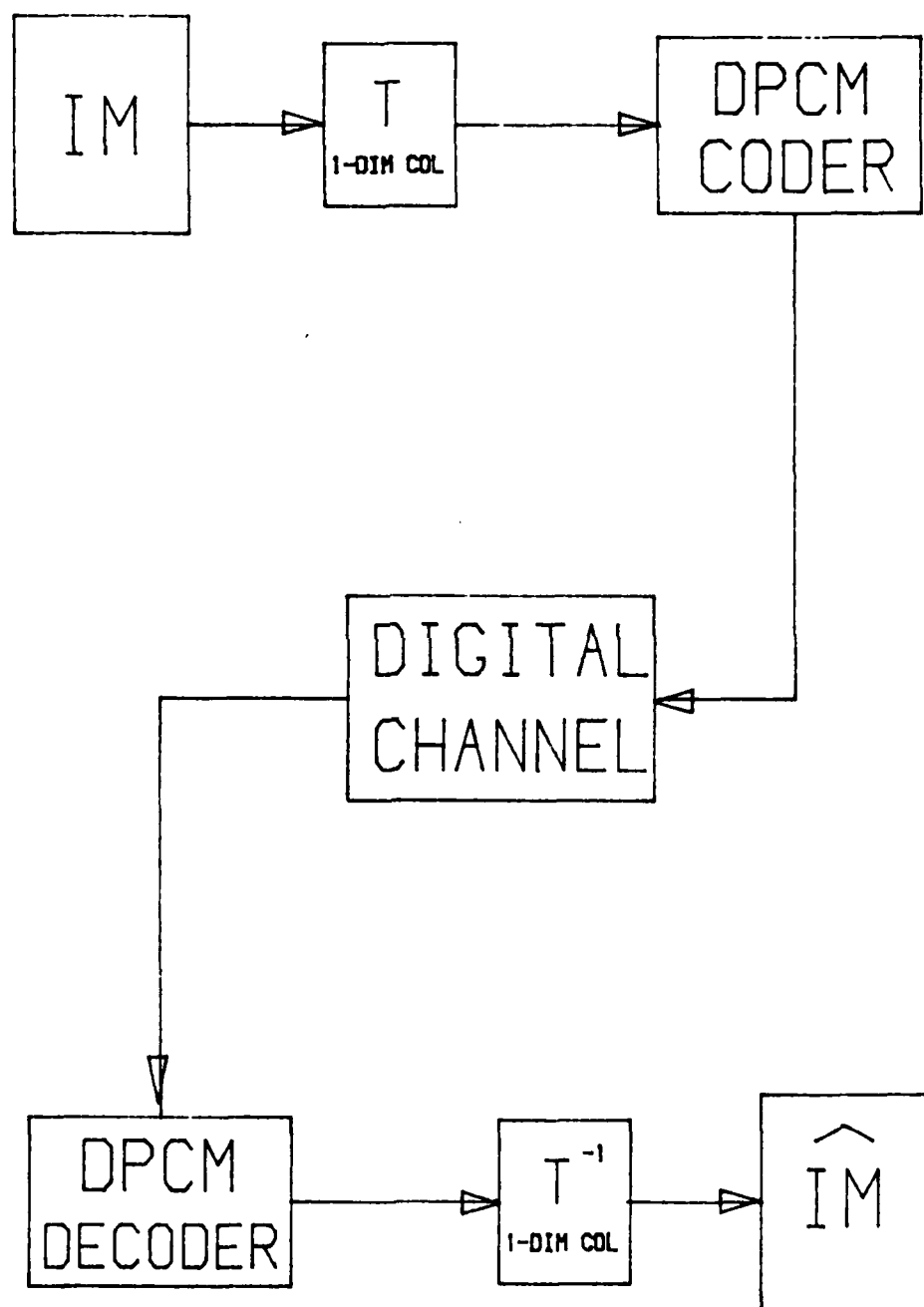
inferior since the original images would not usually all be of this nature, nor even throughout any one image.

More detailed discussions of the transformations may be obtained in Pratt [22]. Of the remaining transformations studied, it appears that the Cosine transform gives the next best performance, irregardless of the block size, followed by the Slant, Hadamard, and Haar transforms. Algorithms of these transforms exist and most are relatively fast.

III. HYBRID SYSTEMS

Both the DPCM and transformation coding systems have their advantages and disadvantages. The DPCM system is relatively easy to implement, but produces an inferior reconstruction as compared to that of transformation coding systems. Unfortunately transformation coding systems involve a more complex design. By combining the two in a particular fashion, it has been discovered that the advantages of both may be realized while minimizing the disadvantages of each system alone.

Habibi [19] proposed that a system be comprised of the two coding schemes in such a manner that a one-dimensional, instead of a two-dimensional, transform be applied to each image block to decorrelate a sequence of the picture data in one direction throughout the block. The



Hybrid Transform/DPCM Coding

Figure 15

transform coefficients are then taken across the block and fed into the input of a DPCM system operating in the orthogonal direction to that of the transform. One dimensional transforms are fast and very easy to implement and combine with the DPCM system to give quick, reliable performance. See figure 15.

As briefly mentioned in the previous chapter, several variations of the original DPCM concept exist, each providing certain advantages in particular applications. One of the most useful, especially in digital image coding, is that of ADPCM (Adaptive Differential Pulse Coded Modulation). With only a small increase in system complexity, the prediction algorithm of the usual DPCM loop can be made to adapt to the statistical variations of the source, producing more accurate predictions for new data and therefore better decorrelation.

The basic idea is to perform an analysis of the data to be predicted in order to arrive at an optimal set of coefficients for the autoregressive predictor. This analysis and subsequent coefficient set are achieved by introducing a learning period into the system. The data is buffered from the ADPCM component for one learning period, causing the system to no longer be real-time, but offset by one learning period. During this time the data is examined by means of such statistical tools as a Kalman filter, ARMA

(AutoRegressive / Moving Average) model, or Stochastic Approximation algorithm.

For a detailed, and thorough, analysis and comparison of the aforementioned three predictors and predictors in general, see the work of my colleague and co-investigator of Hybrid / DPCM systems, Stratigakis [35]. The Kalman filter is well known from Control and Estimation Theory, and its derivation is prolific in the engineering literature [6], [26], [27]. The Kalman filter is the best linear minimum variance estimator and for the case of gaussian distributed signal, noise, and initial state, it is the best of all possible linear and nonlinear estimators [27].

The ARMA model is a method of determining a precise statistical model of the sequence being analyzed, and the estimates of the autoregressive coefficients is obtained by solving a set of equations involving n linear combinations of the estimated autocorrelations of the sequence, known as the Yule-Walker equations [5]. The Stochastic Approximation algorithm is similar to the Kalman filter in form and operation. It is extremely fast but produces an inferior estimation.

Unfortunately these algorithms do not always generate a stable set of coefficients for a particular line. But by making use of the partial autocorrelations of the coefficient set, the stability of the shaping filter can be determined and the coefficient set adjusted if needed. This

process is called PARCOR stabilization. The stabilization and adjustment occur as follows:

- a) The partial autocorrelations of the coefficients (p_i , $i=1,n$) are calculated.
- b) If $p_i > 1$ for any i , then those coefficient values are scaled to some magnitude less than one and the partial autocorrelations are recalculated.
- c) The process iterates until all partial autocorrelations achieve the requirement of their magnitude restriction.

The procedure is therefore to take a row of transformed picture data, apply the coefficient set identification algorithm (Kalman filter, ARMA, or Stochastic Approximation), and stabilize any instabilities in the system by re-adjusting the coefficients to the predictor in the ADPCM loop. Since the transmitter will make its predictions on this set of coefficients, the set must also be made available to the receiver for the ADPCM decoding. It is therefore necessary to quantize and encode this supplemental information into the data stream. Discussion of this quantizer and all other quantizers for the hybrid

system will be presented in Chapter Six.

CHAPTER FOUR

Max - Lloyd Quantizer

Optimization of a quantized representation of signals has produced considerable interest in quantizer research and areas where rate-distortion considerations are of importance. Fundamental to most of the research to date on this problem are the works of Max [25] and Lloyd [24] in their developments of digital transmission systems which minimize quantization error.

The signal to be quantized is assumed to be an ergodic process of known probability distribution. By considering the input signal, S , as composed of an indexed class of sets $\{R_1, \dots, R_N\}$, where:

$$R_i = \{x \mid x_i < x < x_{i+1}, x_1 = -\infty, x_{N+1} = \infty\} \quad (22)$$

to which a corresponding set of representative values $\{y_1, \dots, y_N\}$ is assigned, an index function $\gamma(x)$, $-\infty < x < \infty$, on the partition set $\{R_1, \dots, R_N\}$ is defined as:

$$\gamma(x) = \{i \mid x \in R_i, 1 < i < N\} \quad (23)$$

and a sequence of labels:

$$a(t)_i = \gamma(S_{in,i}(t)) \quad (24)$$

representing the output signal, S_{out} , is generated for transmission.

If a distortion measure on the quantization mapping of partition sets to representative sets is defined as the expected value of some differentiable function $f(\epsilon)$, where ϵ is the quantization error, and the known probability density of the input signal, S_{in} , is $p(x)$, then:

$$\begin{aligned} D &= E [f(S_{in} - S_{out})] \\ &= \sum_{i=1}^N \int_{R_i} f(x - y_i) p(x) dx \end{aligned} \quad (25)$$

To minimize the distortion for a particular fixed value of N , necessary conditions may be obtained by differentiating the distortion (D) with respect to the partition (R_i) and representative (y_i) sets and setting ϵ equal to zero.

Thus:

$$\frac{\partial D}{\partial R_i} = \frac{\partial D}{\partial x_i} = f(x_i - y_{i-1}) p(x_i) \quad (26)$$

$$- f(x_i - y_i) p(x_i) = 0$$

$$i = 2, \dots, N$$

$$\frac{\partial D}{\partial y_i} = - \int_{R_i} f'(x - y_i) p(x) dx \quad (27)$$

$$= - \int_{x_i}^{x_{i+1}} f'(x - y_i) p(x) dx = 0$$

$$i = 1, \dots, N$$

implying:

$$f(x_i - y_{i-1}) = f(x_i - y_i) \quad i = 2, \dots, N \quad (28)$$

$$\int_{x_i}^{x_{i+1}} f'(x - y_i) p(x) dx = 0 \quad i = 1, \dots, N \quad (29)$$

The choice of f is usually a metric function so that:

$$\begin{aligned} f(0) &= 0 \\ f(x) &= f(-x) \end{aligned} \quad (30)$$

If it is required that $Q(x)$ be monotonically increasing, then (28) implies:

$$x_i - y_{i-1} = x_i - y_i \quad i = 2, \dots, N \quad (31)$$

and

$$x_i = \frac{(y_i + y_{i-1})}{2} \quad i = 2, \dots, N \quad (32)$$

Usually the metric function is chosen to be the squared metric:

$$f(x) = x^2 \quad (33)$$

This results in:

$$x_i = \frac{(y_i + y_{i-1})}{2} \quad i=2, \dots, N \quad (34a)$$

or

$$y_i = 2x_i - y_{i-1} \quad i=2, \dots, N \quad (34b)$$

and

$$\int_{x_i}^{x_{i+1}} (x - y_i) p(x) dx = 0 \quad (35)$$

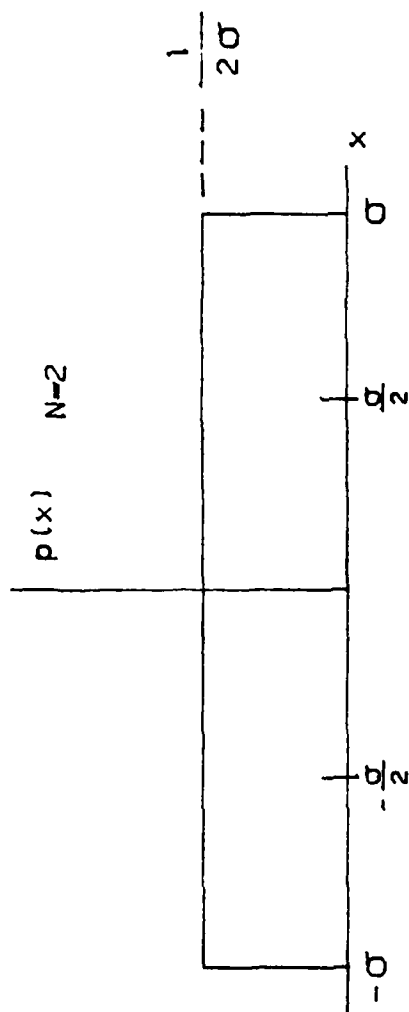
$$i = 1, \dots, N$$

In other words, y_i is the centroid of the area of $p(x)$ between x_i and x_{i+1} .

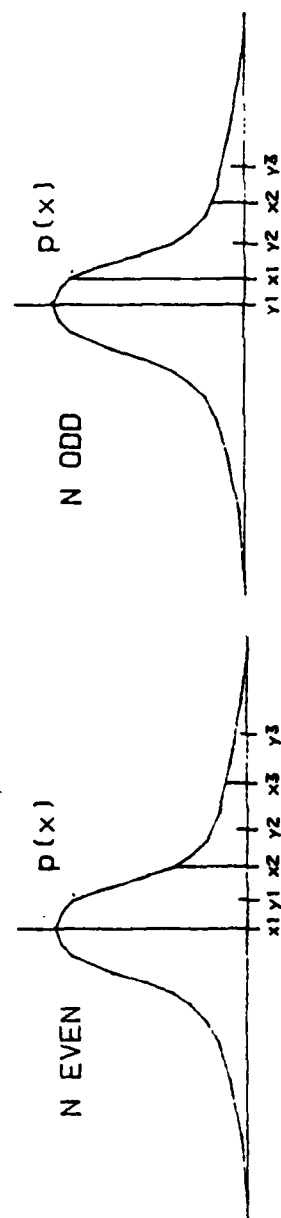
An iterative method may easily be implemented to solve for the optimum, in the mean-squared-error sense, threshold levels $\{x_i\}$ and quantization levels $\{y_i\}$ for the known probability density function, $p(x)$, via equations (34) and (35). Figures of the optimum thresholds and quantization levels for uniform and gaussian probability distributions appear in figure 16.

Once the set of thresholds and quantization levels have been determined, any sample may be quantized by determining the partition set to which it belongs and the representative value for that set by equation (23).

The Max-Lloyd quantizer can now be made adaptive in the same fashion as any other quantizer. A step-size adjustment $\Delta(k)$ is defined such that the thresholds and levels of the quantizer may easily be scaled to some statistical variation estimate. The use of Max-Lloyd



Optimum Quantization for the Uniformly Distributed Case



Optimum Quantization for the Gaussianly Distributed Case

Optimum Quantization

Figure 16

quantizers in an Adaptive Stochastic Picture Coding (ASPC) system for various system parameters and data quantization has been investigated and a detailed discussion of this system and its performance will appear in Chapters Six and Seven.

CHAPTER FIVE

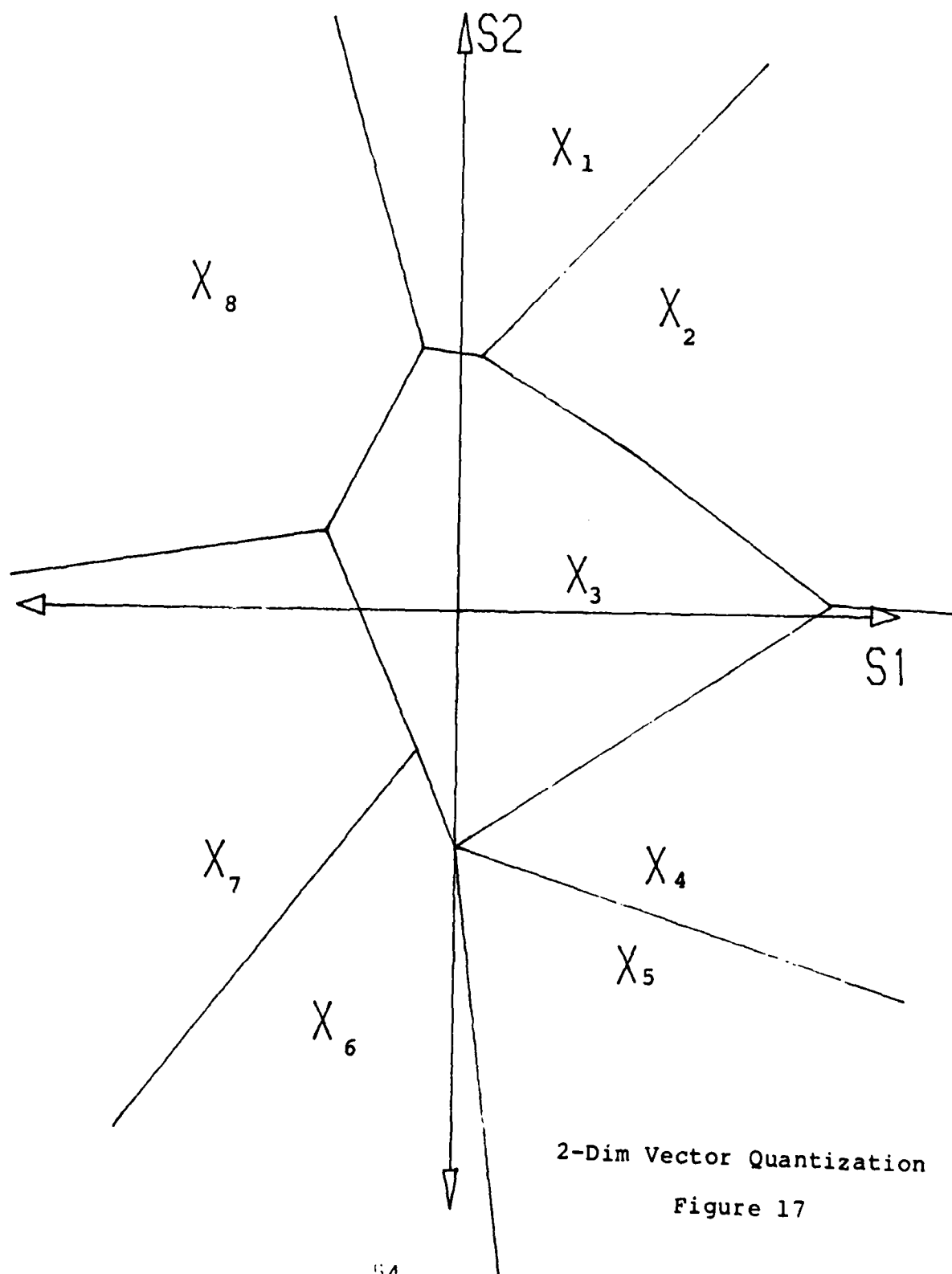
VECTOR QUANTIZER DESIGN

As previously mentioned, quantizing blocks or vectors of samples in lieu of scalar quantization approaches a more accurate representation of the source, in a rate distortion sense, subject only to vector length. By defining vector quantization as a surjective mapping of K-dimensional input vectors to corresponding representative vectors,

$$Q: \overset{K}{R} \longrightarrow \overset{K}{Y}, Y \subset R \quad (36)$$

one can get an intuitive feel for the concept of vector quantization. A typical two-dimensional vector quantizer might appear as in figure 17. Each partition represents a decision rule in the vector space. These partition sets and corresponding representative vectors, X_i , define the vector code book where each X_i is the centroid of the i^{th} partition set.

Since this K-dimensional space must be partitioned into a finite number of regions, it is appropriate to examine the optimality criteria to be achieved to determine the maximal rate distortion performance. Linde, Buzo, and



2-Dim Vector Quantization
Figure 17

Gray [23] have given a fundamental paper on the design of vector quantizer algorithms. Their derivation is presented here since the vector quantizer simulation done for this report was based upon one such algorithm. If an N-level quantizer is defined as optimal (or globally optimal) if it minimizes the expected distortion, then x_Q^* is optimal if for all N-level quantizers x_Q :

$$D(x_Q^*) \leq D(x_Q) \quad (37)$$

If $D(x_Q)$ is only a local minimum, then x_Q is said to be locally optimum.

Likening the distortion measure to that of Max and Lloyd, the measure chosen for this report was the squared-error distortion:

$$d(x, \hat{x}) = \sum_{i=0}^{K-1} |x_i - \hat{x}_i|^2 \quad (38)$$

between x , the sample vector, and \hat{x} , the representation vector. However, as with Max and Lloyd, the derivation is independent of the particular distortion measure chosen.

Given an N-level vector quantizer x_Q defined by the geometric partitioning set $P = \{P_i, i=1, \dots, N\}$ and

constrained representation alphabet $A = \{ y_i ; i=1, \dots, N \}$ then the expected value of the distortion becomes:

$$\begin{aligned} D(\{A, P\}) &= E [d(x, x_i)] \\ &= \sum_{i=1}^N E [d(x, y_i) | x \in P_i] \Pr(x \in P_i) \end{aligned} \quad (39)$$

where $E [d(x, y_i) | x \in P_i]$ is the conditional expected distortion given that $x \in P_i$.

If a particular representation alphabet \hat{A} is given, but a partition is not, then a partition for this alphabet \hat{A} may be found by obtaining the mapping of each x into the y_i \hat{A} minimizing the distortion between the two vectors, $d(x, y_i)$. So the desired partition is found by choosing each vector such that this distortion is a minimum -- this type of clustering is well known to those familiar with pattern recognition techniques as "nearest neighbor" clustering. Anytime the lowest distortion is equal among a number of vectors, any convenient tie-breaking choice may be made. The partition found in this manner yields:

$$x \in P_i^* \iff d(x, y_i) < d(x, y_j) \quad \forall j \quad (40)$$

or

$$D(\{\hat{A}, P(A)\}) \leq E [\min_{y \in \hat{A}} d(x, y)] \quad (41)$$

implying that when compared to any other partition, P:

$$D (\{ \hat{A} , P^* (\hat{A}) \}) \leq D (\{ \hat{A} , P (\hat{A}) \}) \quad (42)$$

So an optimal partition set P^* may be found for any fixed representation alphabet \hat{A} .

Likewise one can determine a best possible representation alphabet \hat{A}^* for any given fixed partition set P. If the distortion measure and distribution are such that each set P with nonzero probability in K-dimensional Euclidean space contains a minimum distortion vector $\hat{x}(P)$ so that:

$$\begin{aligned} E [d(x, \hat{x}(P)) \mid x \in P] \\ = \min_U E [d(x, y) \mid x \in P] \end{aligned} \quad (43)$$

and is termed the centroid of the set P. Therefore no representation alphabet $\hat{A} = \{ y_i; i=1, \dots, N \}$ yields smaller distortion than the alphabet,

$$\hat{x}(P) \triangleq \{ \hat{x}(P_i); i=1, \dots, N \} \quad (44)$$

where each $\hat{x}(P_i)$ is the corresponding centroid of partition P_i since:

$$D(\{A, P\}) = \sum_{i=1}^N E [d(x, y_i) | x \in P_i] Pr(x \in P_i) \quad (45)$$

$$\begin{aligned} &\geq \sum_{i=1}^N \min_U E [d(x, U) | x \in P_i] Pr(x \in P_i) \\ &= D(\{\hat{X}(P), P\}) \end{aligned}$$

Therefore the optimal representation alphabet $A^* = \hat{X}(P)$ for any fixed partition set P .

By using a combination of equations (41) and (45), a method can be implemented for deriving a good quantizer through successive iterations of these constraints given any starting quantizer algorithms using this idea have been proposed for both known and unknown distributions of vectors [23]. Given an initial training sequence of n vectors, an M -level vector quantizer with $M=2^R$, $R=0,1,\dots$ is derived until an initial guess for an N -level quantizer is obtained. The optimal N -level quantizer is then found using this guess. The algorithm chosen for this report work is as follows:

- (1) Initialization: Set $M=1$ and define $\hat{A}_0^*(1) = \hat{x}(A)$ where $\hat{x}(A)$ is the centroid of the training set.
- (2) Given the representation alphabet $A_0(M)$ containing M vectors, $\{y_i, i=1, \dots, M\}$, each codevector y_i is used to produce two code vectors $y_i + \beta$ and $y_i - \beta$ with β a fixed perturbation vector. A new alphabet A of $\{y_i + \beta, y_i - \beta, i=1, \dots, M\}$ with $2M$ vectors is now obtained. Let $M=2M$.
- (3) If $M=N$, then the operation is complete and the alphabet $\hat{A}_0 = \tilde{A}(N)$ is used as the initial reproduction alphabet for the N -level vector quantizer and processing continues with (9). If M is not yet equal to N . then proceed to (4).
- (4) Set $l=0$ and $D = \infty$ and $\text{eps} > 0$ some threshold. Initialize $\hat{A}_0 = \tilde{A}(M)$.
- (5) Given $\hat{A}_l = \{y_i, i=1, \dots, M\}$, find the minimum distortion partition $P(\hat{A}_l) = \{P_i; i=1, \dots, N\}$ of the training sequence:

$$x_j \in P_i \text{ if } d(x_j, y_i) < d(x_j, y_k) \quad k$$

- (6) Find the average distortion for step 1 by:

$$D_1 = D(\{\hat{A}_1, P(\hat{A}_1)\})$$

$$= \frac{\sum_{j=0}^{n-1} \min d(x_j, y)}{n}$$

- (7) If $(D_{l-1} - D_l) / D_l \leq \text{eps}$, then \hat{A}_l is the reproduction alphabet for the M-level quantizer. Set $\hat{A}_0(M) = \hat{A}_l$ and continue at (2). If the $(D_{l-1} - D_l) / D_l > \text{eps}$ continue to (8).

- (8) Find the optimal representation alphabet:

$$\hat{x}(P(\hat{A}_l)) = \{\hat{x}_i(P_i; i=1, \dots, M) \text{ for } P(\hat{A}_l)\}$$

where each $\hat{x}_i(P_i)$ is the centroid of partition P_i . Set $\hat{A}_{l+1} = \hat{x}_i(P(\hat{A}_l))$. Replace $l = l+1$ and continue at (5).

- (9) Now that the initial N-level vector quantizer has been determined, $\hat{A}_0(N)$, the same procedure as outlined in (5) through (8) is performed for an N-level quantizer until the change in distortion between code book calculations drops below a specific threshold or a maximum number of

iterations occurs. The resultant code book should yield a performance within the designated distortion performance.

The scheme in which this algorithm is used will be described in the following chapter on system configurations.

CHAPTER SIX

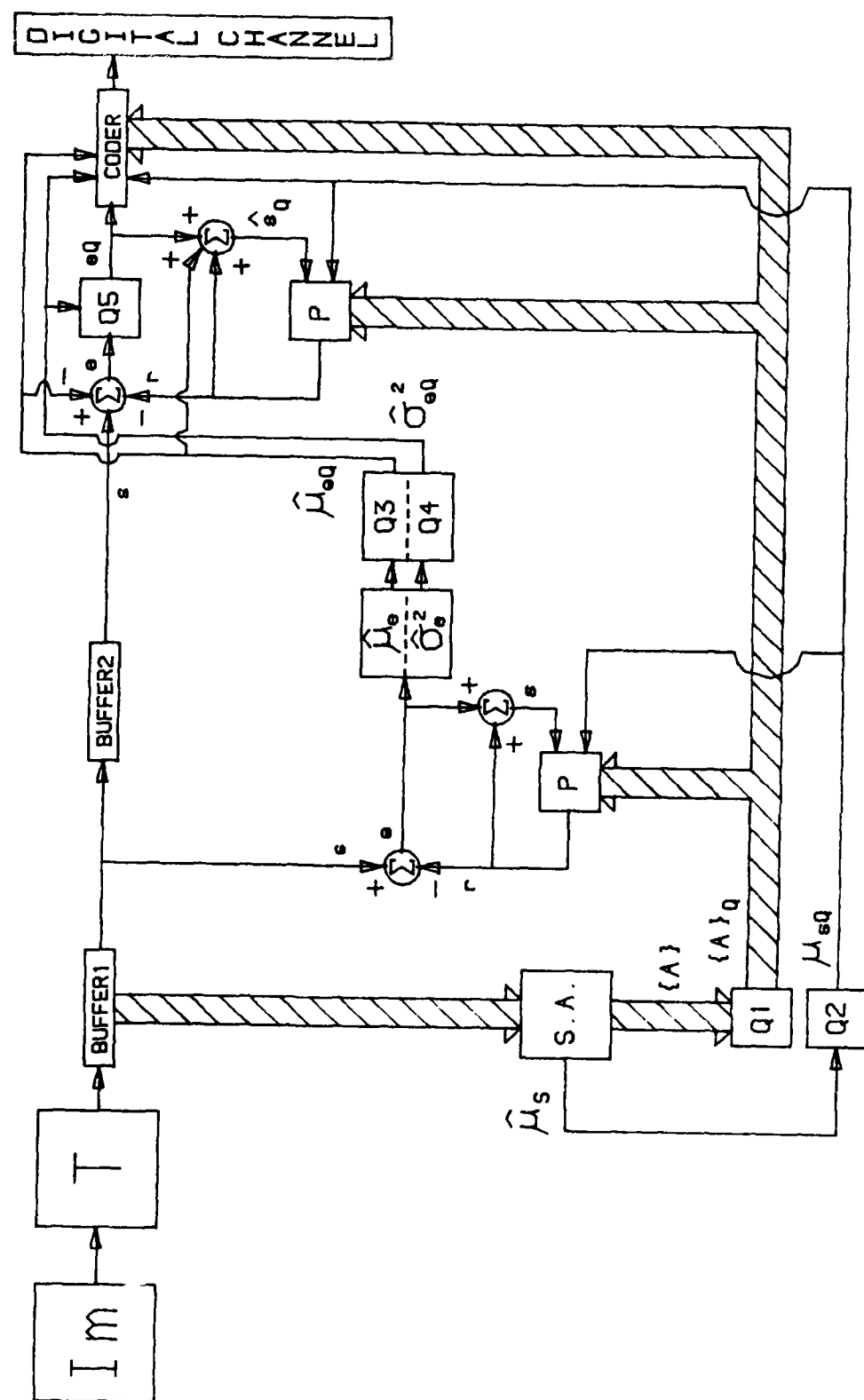
SYSTEM CONFIGURATIONS AND PROGRAM DESCRIPTIONS

I. ASPC / Forward Adaptive Scalar Max

If a good data rate versus distortion performance is to be achieved in a hybrid/DPCM system (also known as Adaptive Hybrid Picture Coding - AHPC - or Adaptive Stochastic Picture Coding - ASPC) it is necessary to examine the transmission requirements for the proper quantizer selections. A block diagram of the proposed transmitter and receiver is shown in figures 18 and 19.

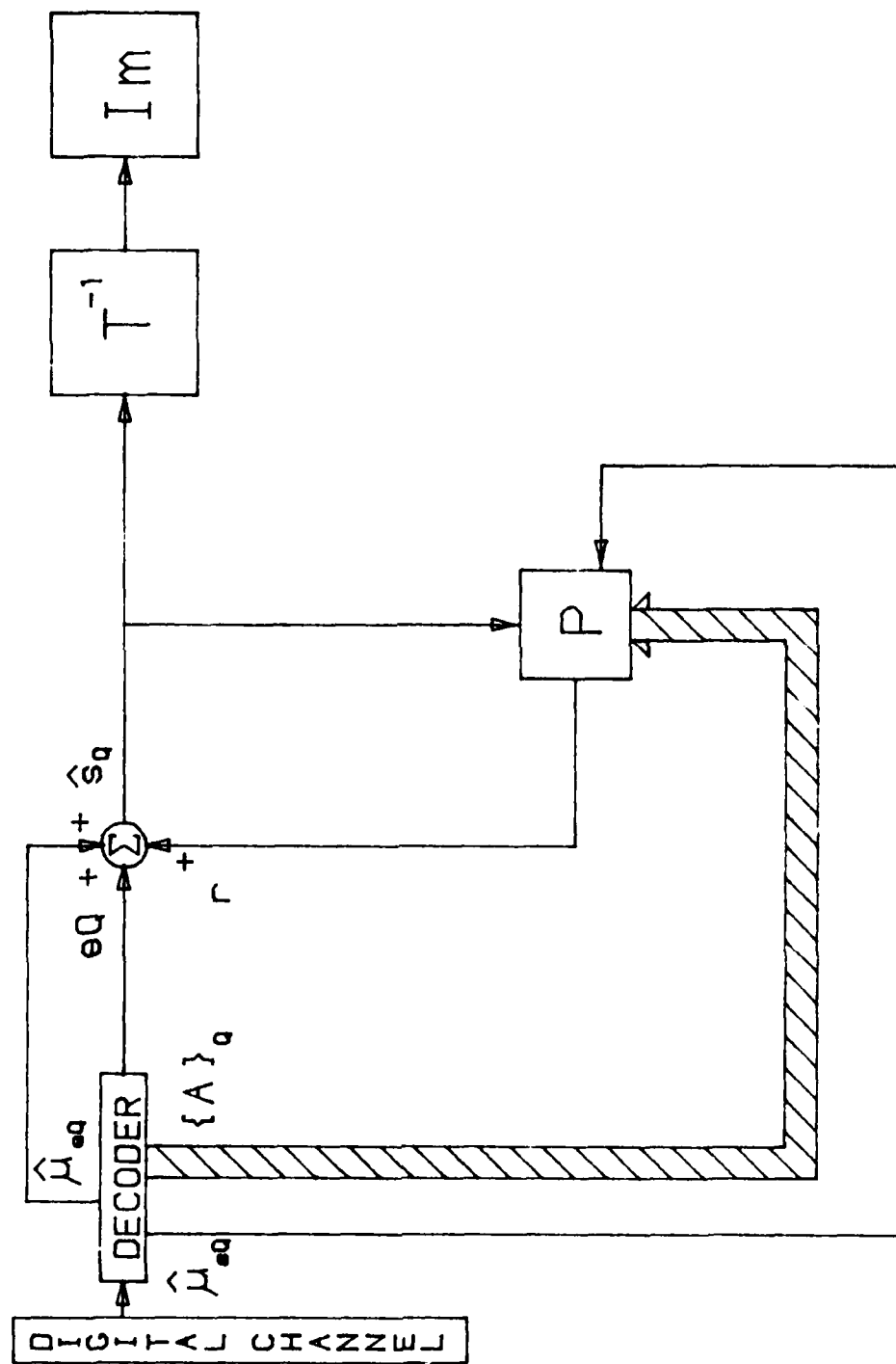
This is the basic design for an AHPC/ASPC system with an autoregressive predictor (P) and a forward adaptive residual Max quantizer (Q5). The operation is as follows:

- 1) The image desired to be transmitted is broken into strips, each of size $BLK \times NS$ where BLK is the desired transform block size and NS is the number of samples in an image line.
- 2) A series of NS columns are passed through a forward unitary transformation for each strip, producing a strip of transformed picture data



ASPC Transmitter with Forward Adaptive Max Quantizer

Figure 18



ASPC Receiver with Forward Adaptive Max Quantizer

Figure 19

which has been decorrelated column-wise.

- 3) Each transformed image line is taken (row-wise) singly and put into the first buffer, BUFFER1.
- 4) This data line is taken in parallel from BUFFER1 and the optimal predictor coefficient set $\{a\}$ for that line is obtained through a Stochastic Adaptation method (Kalman filter, ARMA, Stochastic Approximation) which also produces the mean estimate of the line, $\hat{\mu}_s$. Both the coefficient set $\{a_i\}$, $i=1, \dots, n$, and mean estimate are quantized via Q1 and Q2 for transmission, $\{a_Q\}$ and $\hat{\mu}_{SQ}$, and made available to other parts of the system.
- 5) Since the predictor in the ADPCM loop bases its prediction on n past values contained in an internal shift register, this register must be initialized at the beginning of each line to produce a sensible prediction of the first few values. The quantized signal mean estimate is used for this purpose at both transmitter and receiver stages.
- 6) The values are fed serially from BUFFER1 after the coefficients have been identified for that line. While the system transmits this data line, the next row of transformed image data is

passed to BUFFER1.

- 7) As each value is taken from BUFFER1 it is both passed to another buffer, BUFFER2, and an ADPCM loop with perfect quantization. This loop is necessary in any forward adaptive quantization scheme since a variance or step-size estimate must be obtained prior to quantization. The mean of the residuals is also obtained so that the quantizer may be adjusted to perform about that mean estimate. Since the residuals produced by the Kalman filter and ARMA models should be gaussian, an N-level scalar Max quantizer is used to quantize the residuals in Q5 with an expected gaussian normal probability distribution of mean zero and variance one. The quantizer thresholds and representative levels are all known for this optimum quantizer a priori to both transmitter and receiver. These thresholds and levels are scaled for each line of residuals by a step size determined as the standard deviation estimate (obtained from the quantized variance estimate σ_{eQ}^2). The receiver will need the residual mean and estimate for each line, so only the quantized estimates, μ_{eQ} and σ_{eQ}^2 , are used by the

transmission stage and encoded in the data stream for the receiver. A single variance estimate for each line is assumed sufficient for the statistical variation description of that entire line.

- 8) The values are taken from BUFFER2 and the final ADPCM loop, the residuals of the prediction, e , are acquired and quantized, e_Q , and passed to the coder. The quantized residuals, e_Q , are also added to the prediction, r , so that signal estimates corrupted by quantization noise, s_Q , are obtained for the predictor's n -stage past value shift register.

$$e = s - r - \hat{\mu}_{SQ} \quad (46)$$

$$e_Q = Q(e) = e + q_n$$

$$s_Q = e_Q + r + \hat{\mu}_{SQ}$$

$$= s + q_n$$

- 9) All pertinent data required by the receiver for proper reconstruction of the image is then encoded and passed through a digital channel to the receiver.

- 10) At the receiver stage, the information is decoded into its various components. The quantized signal mean estimate, $\hat{\mu}_{SQ}$, is used to initialize an n-stage shift register in a predictor here, and the quantized coefficient set for this data line is provided to the predictor.
- 11) Each prediction value, r , is added to the received quantized residual, e , and adjusted to the appropriate mean to obtain the signal estimate of each transformed pixel.

$$s_Q = e + q_n + \hat{\mu}_{SQ} = s + q_n \quad (47)$$

- 12) These values are used to rebuild the original transform-domain strips.
- 13) The strips are finally passed through an inverse column transformation and pieced together to reconstruct the original image.

Besides the quantizer used for the residual sequence, Q5, four other quantizers were needed for the various system parameters, none of which were to be adaptive. The quantizers were uniquely designed for each parameter. The quantizer for the coefficient set, Q1, was

designed as an 8-bit uniform quantizer of the limits of ± 1.75 . These values as were all quantizer parameters for quantizers Q1-Q4 were determined initially by estimation and finalized through experimentation.

The quantizer for the signal mean, Q2. was originally designed as a 9-bit fixed Max-gaussian quantizer which used two sets of parameters. One set was used with those image rows occurring at the beginning of each block of transformed values since such rows contained the highest energy - and thus more widely variant - transform coefficients. The values used for this set were $128 \times \text{BLOCKSIZE}$ for the mean estimate and a variance estimate of about 130,000. The estimation of $128 \times \text{BLOCKSIZE}$ for the signal mean came from the median of the 256 possible gray levels in a monochromatic image (128) summed over the block. The second set was used for all remaining data lines of the block and consisted of a zero mean estimate and a variance estimate of 350. It was later determined that for a particular transformation, these signal mean values could be approximated to eliminate the necessity of quantizing and transmitting this parameter (the approximation being known a priori to both transmitter and receiver) to actually improve the system's performance. In the case of the Cosine transformation, signal mean estimates of 2750 for lines at the beginning of a block and 0 for all other lines were used. This elimination of the need to transmit the signal

mean gave a data rate savings of 9 bits per line. These values would, of course be different for other transformations. Exact system data rates will be calculated in the next chapter.

The residual mean estimate was assumed gaussian over the entire image and thus a time invariant Max quantizer, Q3, was used here. It was of 6-bits in resolution with mean of zero and variance of 150. The residual variance estimate quantizer, Q4, was uniform from 0 to 40,000 for the ARMA and Kalman filter estimators, and 0 to 100,000 for the Stochastic Approximation estimator. Since this parameter yielded the step-size estimate for the adaptive residual quantizer, a resolution of 9 bits was used. The reason for the wider distribution of the variance for the Stochastic Approximation routine is that it performs only a quick estimation of the predictor coefficients, thereby causing the generated residual sequence to be of a more variant nature.

II. ASPC / Backward Adaptive Scalar Max

As described before, a system can be used such that no extra quantizer information, such as a step-size adjustment, is needed. This system would use the backward adaptive quantizer which acquires its step-size adjustment

from previously quantized values readily available at both transmitter and receiver stages.

It was seen in Chapter Two that there are as many forms of step-size algorithms for the backward adaptive quantizer as for the forward adaptive quantizer. The one selected for this study was that of equation (4), reprinted here for convenience:

$$\Delta(k) = \frac{1 - \alpha_1}{\alpha_2} \sum_{i=1}^{\infty} \alpha_1^{i-1} |eQ(k-i)| \quad (48)$$

This particular formula for the step-size was chosen for its similarity to the autoregressive prediction being performed in the ADPCM loop. Choices of scalars α_1 and α_2 are determined experimentally for particular system. The simulation performed for this report used values of $\alpha_1 = 0.7$ and $\alpha_2 = 1$. Future inquiries into this algorithm could possibly discover more suitable parameters, but these were judged significant and typical for this comparison of quantization schemes.

With a backward adaptive quantizer, the transmitter of figure 18 is modified somewhat since a prior residual variance is no longer needed. However the residual mean is still required for proper orientation of the residual quantizer. The system does require additional circuitry since an estimate must be made from previous quantized

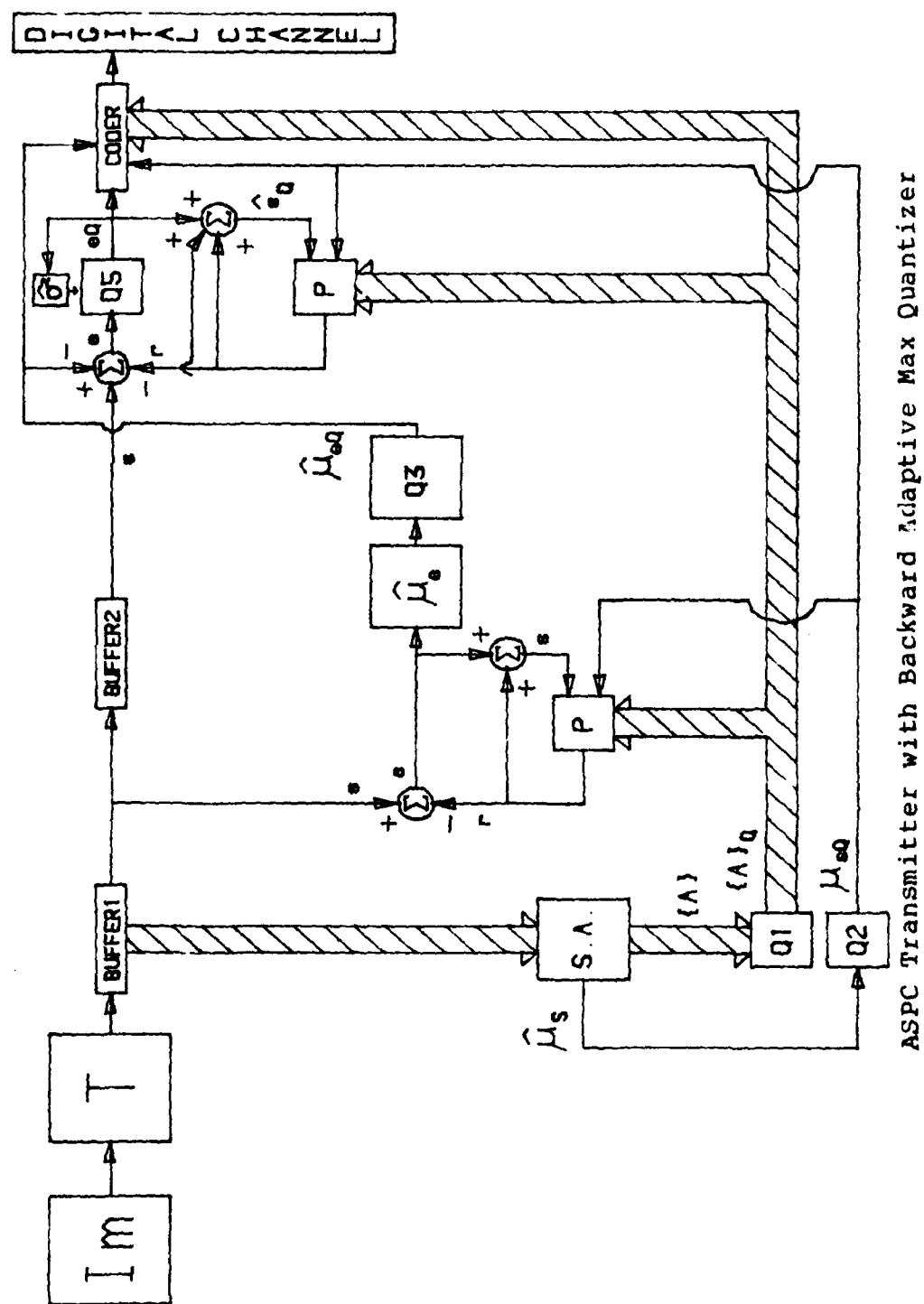
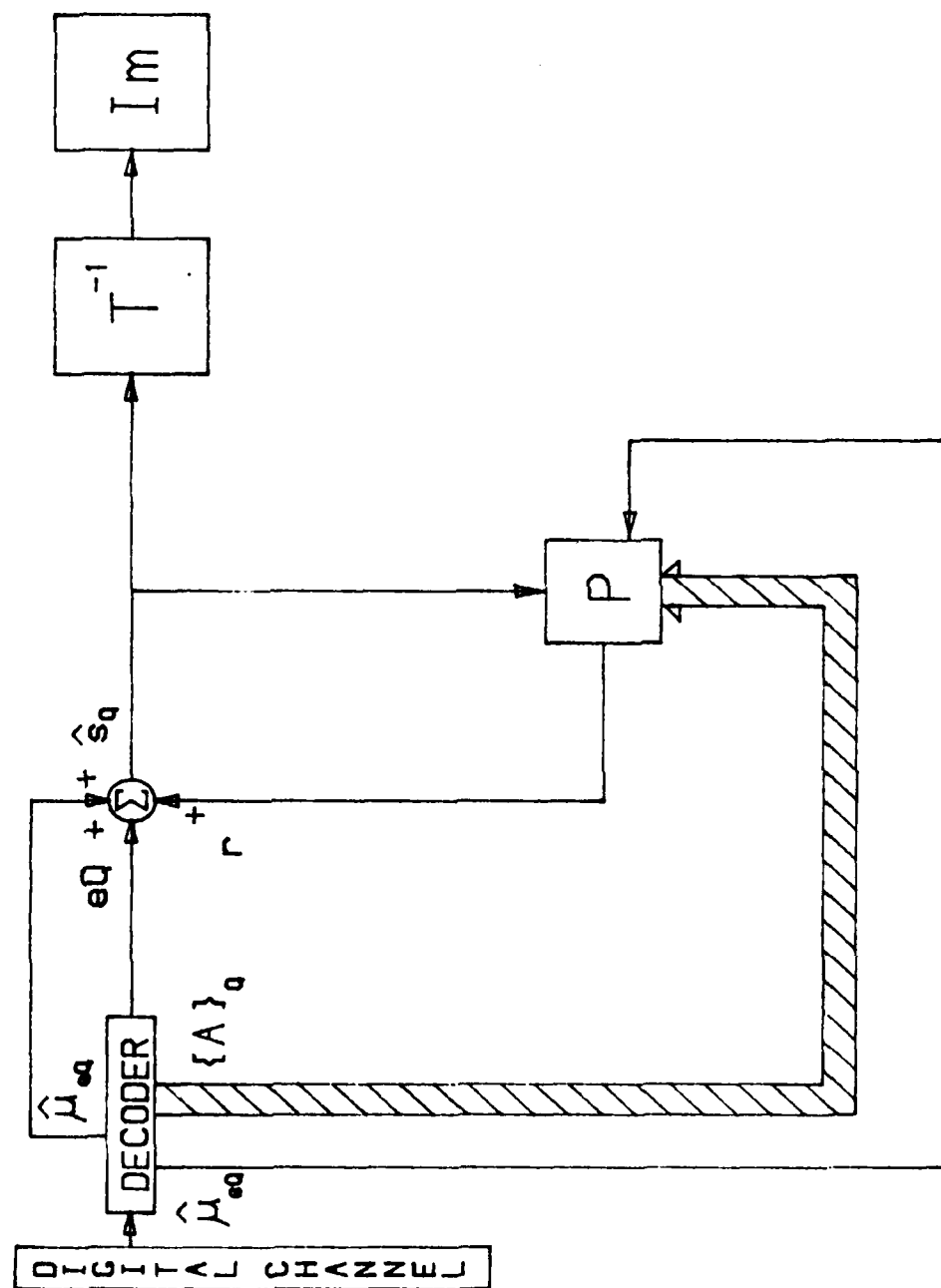


Figure 20



ASPC Receiver with Backward Adaptive Max Quantizer

Figure 21

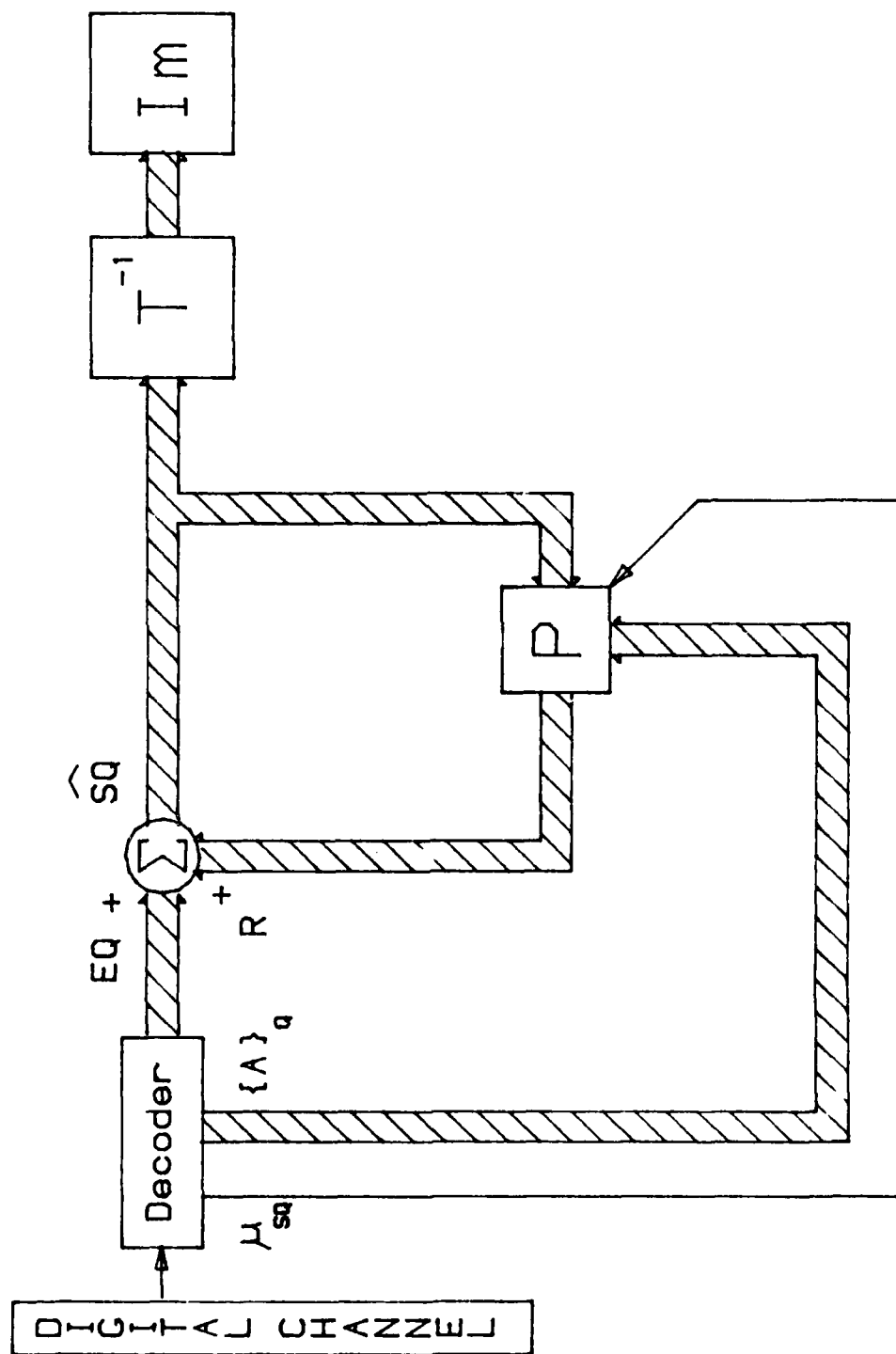
values. The resulting transmitter block diagram is depicted in figure 20. The receiver block diagram remains as shown in figure 21.

III. ASPC / Vector Quantization

An extension of the system using scalar quantization techniques of figures 18 and 19 results in the vector AHPC system, called VASPC, as depicted in figures 22 and 23. The operation of the VASPC system is nearly identical with that of the ASPC system -- with the obvious exception of the vector quantization.

The original image is still split into strips of the transformation blocksize by the image width and forward transformed columnwise. But now instead of applying ADPCM on a single line at a time, the system has the ability to handle any number of lines at once. Since the usual image column length is a multiple of 2, and the transformation blocklength also a multiple of 2, say 16, the vector dimensions examined are $K = 2, 4, 8, 16$. It must be kept in mind that any practical size can be specified, but any additions in dimension increases the corresponding vector code book / alphabet as well as system complexity.

The coefficient set for each line is determined in precisely the same fashion as the scalar AHPC case. The



Vector ASPC Receiver

Figure 23

first vector ADPCM loop figured in the diagram serves another purpose than that of the scalar case. Recall that previously this particular loop was for the determination of an estimate of the residual variance of each transformed image line. In the VAHPC system this loop must aid in determining the whole vector quantizer code book. K lines are operated on at once in parallel and are considered to be NS (number of samples per line) vectors in the Kth dimension. A corresponding predicted K-dimensional vector is produced for each, and the resultant residual vectors are collected to form a training set for the design of the vector quantizer.

$$\underline{e}_i = \underline{S}_i - \underline{R}_i \quad i=1, \dots, NS \quad (49)$$

$$\underline{E} = \{ \underline{e}_i \}_{i=1, \dots, NS} \quad (50)$$

This set of training vectors is then fed into the design algorithm to produce a quantizer for this strip of K X NS residuals. Since this code book must be available to the receiver, a mean estimate vector and a variance estimate vector for this code book are generated, and a K-dimensional Max gaussian quantizer scaled to these statistical vectors is employed to quantize and transmit the code book to the

receiver. For small vector dimensions the scheme of design and transmission could yield an enormous overhead if it is done for every NSF (number of image lines) / K size strip. Instead, after each code book is designed, each codevector is compared with each codevector of the previous K X NS strip code book. A distortion measure is obtained from this comparison, and the present code book is shuffled such that those vectors achieve a minimum distortion. Those falling below a specific threshold are not transmitted. No vector code book is necessarily similar to the previous one, and therefore a vector which need not be transmitted does not necessarily have the same index as the vector it resembles of the previous code book. Because of this fact, it is necessary to give an index to the old code book for which a vector is being replaced. This indexing itself constitutes an overhead, so a decision must be made as to whether it is better to transmit a whole new code book, a partial code book, or retain the previous code book. This decision also causes a small 2 bit / code book overhead, but is unchanging throughout the picture. This decision takes the form of:

$$\begin{aligned}
 K * NBE * 2^B &= B * \text{NUMVTR} + K * NBE * \text{NUMVTR} & (51) \\
 &= (B + K * NBE) * \text{NUMVTR}
 \end{aligned}$$

where:

K = Vector dimension

NBE = Number of bits used to quantize each
element in each vector.

NUMVTR = Number of vectors needing to be
transmitted .

So for a 4 bit vector quantizer, B=4 (16 codevectors) and
the decision becomes:

<u>DIM</u>	<u>Transmit whole book if:</u>
2	NUMVTR > 12
4	NUMVTR > 13
8	NUMVTR > 14
16	NUMVTR > 15

and for a 16-dimensional quantizer, a partial code book is
transmitted even if only one vector is similiar to a
previous vector. If the whole book is to be transmitted,
the indexing is dropped. For a comprehensive data rate
calculation, refer to Chapter Seven.

IV. ASPC / Forward Adaptive Scalar Max with "Zeroing"

The last configuration considered is actually a
degenerative form of the first ASPC configuration with a

forward adaptive scalar Max gaussian residual quantizer. This system was simulated only to give an approximation of AHPC performance at non-integer bit rates (neglecting overhead). Since there is no way to design a Max quantizer with 0.5, 1.5, 2.5 or any non-integer bit resolution, a method called conditional-replenishment or "zeroing" is used to get a more generalized picture of the scalar AHPC rate distortion performance. Even if one could make such a 1.5 bit quantizer how would one transmit 1.5 bits?

Since only a performance estimate is desired, it is suitable to design for an effective performance. The procedure is simply to not transmit portions of the residual sequence, thus reducing the data rate. So a residual of zero, or a perfect prediction is assumed, and the subsequent error in the next prediction is expected to be corrected in the residual, which is transmitted. Residual sequences might then appear:

```

e 0 e 0 e 0 ...
 1   3   5
e 0 0 e 0 0 e 0 ...
 1       4       7
e 0 0 0 e 0 0 0 e 0 ...
 1           5           9
      etc.

```

The effective bit rate is then:

$$\frac{B}{\text{Number of sharing res.}} = \frac{\frac{\text{Log } N \text{ (\# bits in res. qntzer)}}{2}}{\text{Number of sharing residuals}}$$

Of course the performance is quite inferior at all data rates to the normal AHPC system since the coefficient identification routines are designed for a different sequence. Instead of the n-stage past value shift register containing samples perturbed only by quantization noise:

$$\hat{s}_Q = e_Q + r = e + r + q_n \quad (52)$$

there is the additional burden of a previous prediction error:

$$\hat{s}'_Q = e_Q + r + e' = e + r + q_n + e' \quad (53)$$

and the prediction scheme uses coefficients which have not been optimized for this sequence. But the routine still yields a general "feel" of the scalar AHPC rate distortion performance at odd data rates.

V. Program Descriptions

The computer programs used for the simulations presented here are slightly lengthy but have been designed in as modular a fashion as possible to achieve a flexibility in design changes and alternatives.

The transformation program consists of a driving program and a set of transform subroutines. The image to be used is selected from the disk on which it has been stored, and provided with the desired transformation, blocksize, and transformation direction(s) (row or column), the image is transformed and the resultant transformed image is placed on disk. All disk operations are performed with a subroutine called DISKIO which handles the image files a line at a time in any desired manner.

The main transmitter / receiver program consists of a small main driver segment to read the original transformed image and write the received and reconstituted transformed image to disk. The main subroutine is called COMPMC which performs the simulation of ADPCM transmission, reception, and reconstruction. Minor changes are introduced into each run of COMPMC which performs the simulation of ADPCM transmission, reception, and reconstruction of the transformed image data. Minor changes are introduced into each run of COMPMC for the specific coefficient prediction schemes and all scalar quantizers.

Subroutines which are also required include:

MAXQTZ - subroutine for designing any N-level scalar Max / Lloyd quantizer using the techniques derived by Max.

PARSBL - subroutine for the partial auto-correlation stabilization of the predictor coefficients.

CENTRD - subroutine for finding the centroid level in a probability distribution between two thresholds for MAXQTZ.

INTGRL - a 16-point gaussian quadrature integration subroutine required by both CENTRD and MAXQTZ.

Functions used are:

UNIFRM - function to provide an N-level uniformly distributed quantization between two specified intervals.

FUNC - function to generate a desired probability density function.

SQERR - function to generate the squared error distortion of a value in a particular probability distribution.

QNTZ - function to quantize a sample via Max's definition given a variance estimate and

the quantization thresholds and levels produced by MAXQTZ. The step-size scale factor is determined as the square-root of the variance estimate (std deviation).

The VASPC vector program uses a subroutine called COMPV quite similar to COMPMC. Special provision had to be made for the vector quantizer but the general flow of the routine was the same. Besides the aforementioned subroutines, COMPV requires:

VQDSN - subroutine for random vector quantizer design based on a given training set.

VECQNT - subroutine to determine the representative or quantized value of a vector via the "nearest neighbor" approach.

COMPMC and COMPV both require subroutines for the desired coefficient identification routine, with the exception of the Kalman filter routine which had been directly incorporated into one version each of COMPMC and COMPV. Other identification routines require:

FTAUTO - IMSL subroutine for finding the auto-correlations of a given sequence.

FTARPS - IMSL subroutine for generating a set of

coefficients for a designated Auto-Regressive Moving Average (ARMA) model given the autocorrelation produced by FTAUTO.

STAPRX - subroutine for determining the prediction coefficients via the Stochastic Approximation algorithm.

Finally, the received and reconstructed transformed images are inverse transformed with the same original transform program package and the final image is placed on disk. Programs were also used to calculate the signal-to-noise ratio between the original and final images and generate the corresponding error histograms. These programs are SNR and HIST1, respectively. General purpose routines were used for the transfer of images from disk to tape and vice versa for subjective viewing on the COMTAL Vision One Image Processing system of the Visual Interactive Processing laboratory, a part of the University of Arkansas' Engineering Experiment Station. Flowcharts and program listings are included in Appendix B, and the reader is referred to this appendix for further information on the details of the simulations.

CHAPTER SEVEN

RESULTS AND CONCLUSIONS

A number of orthogonal transformations exist which could have been used in the simulations, but the aim of this study was not the comparison of transforms for AHPC. The simulations mentioned in Chapter Six were performed for two types of transforms: the Hadamard and the Cosine. These two were chosen for their frequent use in other work in this area of research and widespread use in current scientific publications. Further, the Hadamard represents a transform of average complexity and performance, while the Cosine's performance appears to excel in transformation coding systems [22]. Finally these were the two transforms used by Habibi [19] in his original work on Hybrid / DPCM systems. Only the Cosine was used after it was determined experimentally to perform better for AHPC systems than the Hadamard for any coefficient identification scheme.

Two images were used as originals in the initial stages of the system simulations. These are the 128 X 128 monochrome image of the well-known MIT girl used in much of the literature, and the 256 X 256 monochrome image of an aerial photograph of a city. See pictures <1> and <2> in

[illegible]

ASPC SNR TABLE (dB)

Figure 24

PREDICTOR PERFORMANCE

DIAMOND - KALMAN FILTER
STAR - ARMA MODEL
PLUS - STOCH. APPROX.

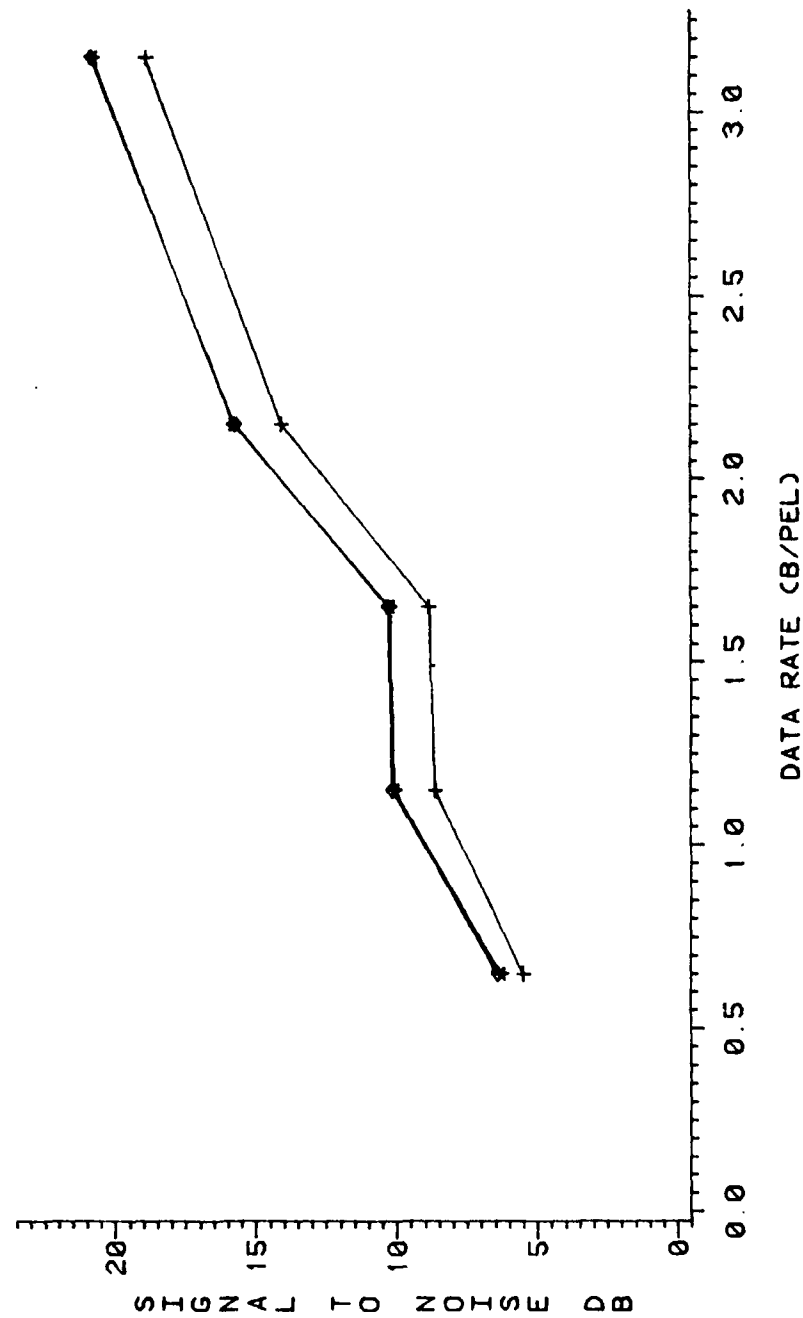


Figure 25

Appendix A. ASPC simulations were performed on each image with both transformations and with and without PARCOR stabilization to determine the best overall coefficient identification / transform scheme to use in the quantizer comparison. It was determined that the best overall performance was given for the cosine transformed city image with a set of 3 predictor coefficients, controlled with PARCOR stabilization. See figure 24. All ASPC simulations on this table were performed with a transform block size of 16, a 3-bit forward adaptive residual Max gaussian quantizer, an 8-bit uniform coefficient quantizer, a 6-bit Max gaussian residual mean estimate quantizer, a 9-bit uniform residual variance estimate quantizer, and a 9-bit Max gaussian signal mean quantizer. These yield fairly high data rates, but a good objective viewpoint of system performance.

The cosine-transformed city image was then used in determining the best overall predictor performance. As seen from figure 25, the Kalman filter and ARMA model resulted in almost identical rate distortion performance and were significantly superior to the Stochastic Approximation algorithm. The Kalman filter was then chosen to perform the coefficient identification for the quantizer comparisons. The Kalman filter adaptive identification process is

AD-A138 876

ADAPTIVE HYBRID PICTURE CODING(U) ARKANSAS UNIV
FAYETTEVILLE DEPT OF ELECTRICAL ENGINEERING
R A JONES ET AL. 30 NOV 83 AFOSR-TR-84-0142

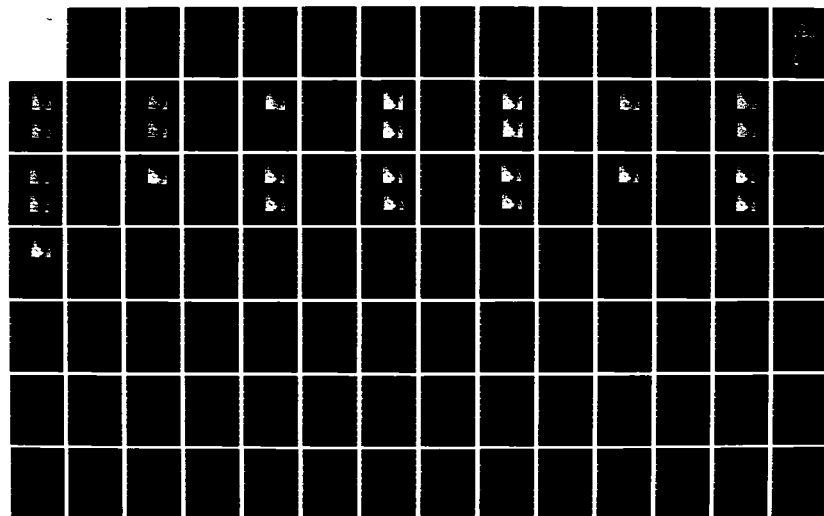
2/3

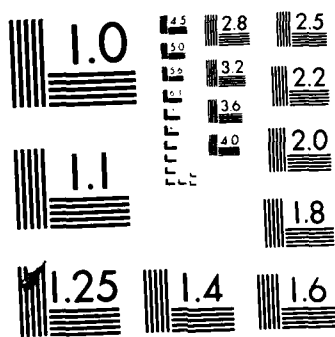
UNCLASSIFIED

AFOSR-82-0351

F/G 12/1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

summarized [21]:

$$\text{Message Model} \quad A(k+1) = A(k) \quad (53)$$

$$\text{Observation Model} \quad s(k+1) = R^T(k) A(k) + e(k+1)$$

$$\text{Identification eqn} \quad \hat{A}(k+1) = \hat{A}^p(k) + K(k+1) e(k+1)$$

$$\text{Gain equation} \quad K(k+1) = \frac{V_{\hat{A}}(k) R^T(k)}{V_Z + R^T(k) V_{\hat{A}}(k) R(k)}$$

$$\text{Variance eqn} \quad V_{\hat{A}}(k+1) = [I - K(k+1) R^T(k)] \cdot V_{\hat{A}}^p(k)$$

where at time k:

$A(k)$ - Coefficient vector

$s(k)$ - sample value

$e(k)$ - residual value

$R_p^T(k)$ - past value vector

$R^T A(k)$ - predicted value

$K(k)$ - Kalman gain

V_Z - error variance = 100

$V_{\hat{A}}(k)$ - a posteriori error variance

A graph depicting the performance of each quantizer routine is shown in figure 26. The corresponding images appear in pictures < 3 - 27>. As predicted by the theory, the vector quantizer produces the best overall subjective and quantitative performance, followed closely by the forward adaptive scalar version and a quite inferior

AHPC QUANTIZER PERFORMANCE

KALMAN FILTER PREDICTOR
 STAR - VECTOR MODEL
 DIAMOND - FORWARD ADAPTIVE MAX
 PLUS - BACKWARDS ADAPTIVE MAX

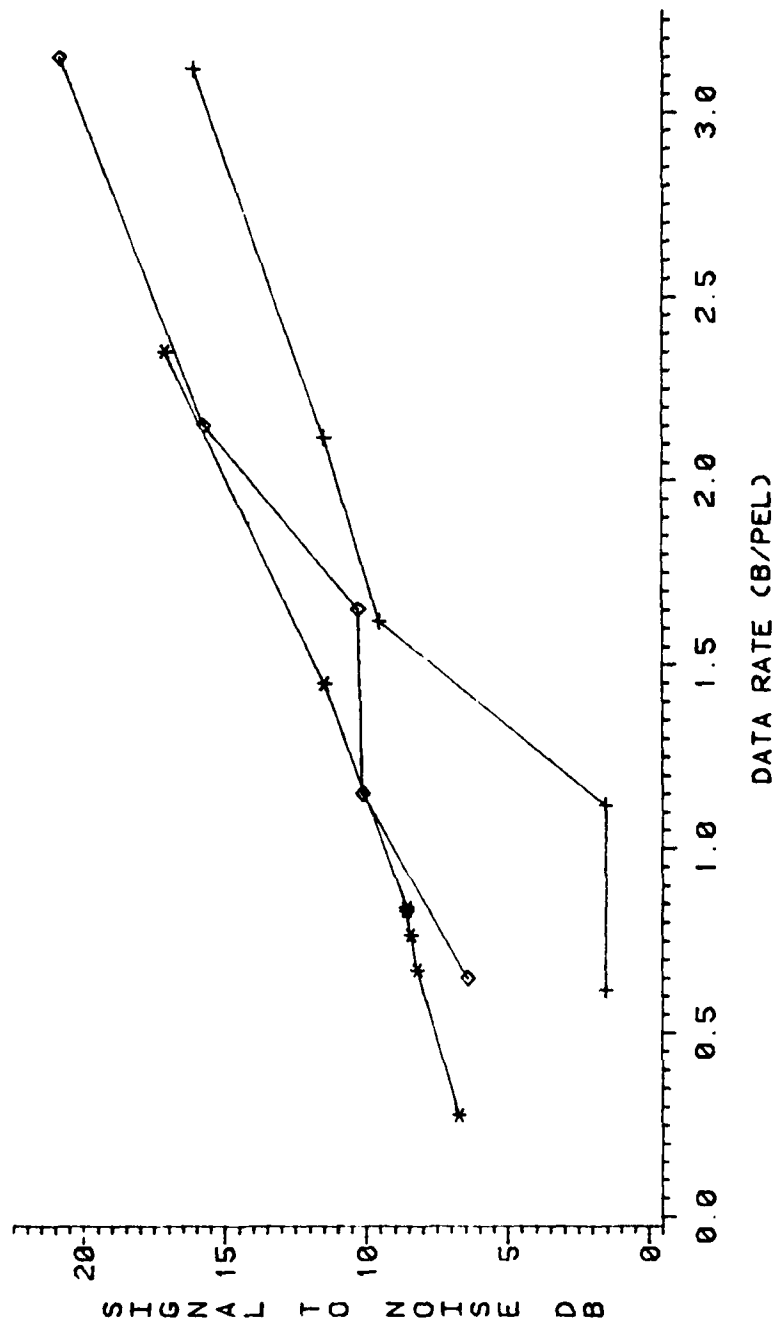


Figure 26

backward adaptive scalar version.

Although there does not appear to be a consensus on the calculation of a signal-to-noise ratio performance for images -- which would greatly aid the comparisons and evaluations of new and different techniques -- a ratio of signal variance to noise variance was arrived at as a good critical evaluation. The data rate calculations are as follow:

Forward Adaptive Scalar Max Residual:

Coeff. reqs.	256 lines * 3 coef/line * 8 b/coef
Res. Var.	256 lines * 9 b/line
Res. Mean	256 lines * 6 b/line
Res. Qntzr	R bits/pel

=> R b/pel + 0.15 b/pel overhead

Backward Adaptive Scalar Max Residual:

Coeff. reqs.	256 lines * 3 coef/line * 8 b/coeff
Res. Mean	256 lines * 6 b/line
Res. Qntzr	R bits/pel

=> R b/pel + 0.12 b/pel overhead

Vector Quantizer:

No Code book transmission

$2 * \text{Number of no transmissions (decision)}$

Whole Code book transmission

$2 * \text{Number of full transmissions (decision)}$

$\log_2 N * K * 7b/\text{elem} * \text{Number full trans.}$

$K * 9\text{bits} * \text{Number full trans. (variance)}$

$K * 6\text{bits} * \text{Number full trans. (mean)}$

Partial Code book transmission

$2 * \text{Number of partial transmissions (decision)}$

$\log_2 N * K * 7b/\text{elem} * \text{Number part. trans.}$

$K * 9\text{bits} * \text{Number part. trans. (variance)}$

$K * 6\text{bits} * \text{Number part. trans. (mean)}$

==> Actual data rate must be calculated
interactively during system simulation.

These systems show the feasibility of various digital image coding and transmission schemes. Vector quantization, although a relatively new technique, appears to have great promise in any such data reduction system. As seen here the AHPC system can yield good results at low data rates. Pictures and error histograms supporting this conclusion appear in Appendix A. All signal-to-noise ratio

calculations were made as a ratio of the variances:

$$\text{SNR} = 10 * \text{Log}_{10} \frac{\sum_{i=1}^N \sum_{j=1}^N (s_{ij} - \bar{s})^2}{\sum_{i=1}^N \sum_{j=1}^N e_{ij}^2} \quad (\text{dB}) \quad (54)$$

Future Research

The AHPC system has been shown to give good rate distortion performance. The limiting factor has also been shown to be the quantizer performance. Optimization of any initial parameters such as those used in the identification routines may improve performance, as well as determining any better step-size estimators. Any future research endeavor into AHPC-type systems should therefore include any new thoughts as to the possible evolution of quantization theory. Vector quantization research appears the most lucrative for low data rate performances. Although other forward adaptive and backward adaptive quantizer algorithms may exist which yield better results than those presented here, these are figured typical and any such scheme appears limited by the one-dimensional scalar quantization in

adaptive systems.

One possibility of future research efforts is the incorporation of the AHPC system into a real-time transmission of sequential images. Special quantization techniques may be employed in such systems which take advantage of previous image frame information. Considerations of any other interframe knowledge could improve an intraframe performance of AHPC.

A problem which might also be scrutinized is that of an introduction of noise into the system. What are the effects of channel errors on the receiver's reconstructed image? How does initial noise in the original image and noise in the image in the transform domain affect system reliability? Perhaps various filtering techniques could be applied at points in the system to limit image deterioration. Only more research can provide the answers to these questions.

REFERENCES

1. Arnstein, D.S., "Quantization Error in Predictive Coders," IEEE Transactions on Communications, Volume COM-23, April 1975, pp. 423-429.
2. Bennett, W.R., "Spectra of Quantized Signals," Bell System Technical Journal, Volume 27, July 1948, pp. 446-472.
3. Berger, T., Rate Distortion Theory, Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.
4. Berger, T., "Minimum Entropy Quantizers and Permutation Codes," IEEE Transactions on Information Theory, Volume IT-28, Number 2, March 1982, pp. 149-157.
5. Box, George E. P., and Jenkins, Gwilym M., Time Series Analysis: Forecasting and Control, revised edition, Holden-Day Inc., San Francisco, CA, 1976.
6. Brown, Robert G., Introduction to Random Signal Quantization and Kalman Filtering, John Wiley & Sons Inc., New York, NY, 1983.
7. Bucklew, James A., "Companding and Random Quantization in Several Dimensions," IEEE Transactions on Information Theory, Volume IT-27, Number 2, March 1981, pp. 207-211.
8. Buzo, A., Gray, A. H., Gray, R. M., and Markel, J. D., "Speech Coding Based upon Vector Quantization," IEEE Transactions on Acoustics, Speech, and Signal Processing, Volume ASSP-28, October 1980, pp. 562-574.
9. Cohn, D. L., and Melsa, J. L., "The Residual Encoder - An Improved ADPCM System for Speech Digitization," IEEE Transactions on Communications, Volume COM-23, Number 9, September 75, pp. 935-941.
10. Cohn, D. L., and Melsa, J. L., "A Pitch Compensating Quantizer," Conf. Rec. 1976 IEEE Int. Conf. ASSP, 1976, pp. 258-261.

11. Gersho, A., "Principles of Quantization," IEEE Transactions on Circuits and Systems, Volume CAS-25, Number 7, July 1978, pp. 423-436.
12. Gersho, A., "Asymptotically Optimal Block Quantization," IEEE Transactions on Information Theory, Volume IT-25, Number 4, July 1979, pp. 373-380.
13. Gersho, A., "On the Structure of Vector Quantizers," IEEE Transactions on Information Theory, Volume IT-28, Number 2, March 1982, pp. 157-166.
14. Gibson, Jerry D., Jones, Stephen K., and Melsa, James, "Sequentially Adaptive Prediction and Coding of Speech Signals," IEEE Transactions on Communications, Volume COM-22, Number 11, November 1974, pp. 1789-1797.
15. Gibson, Jerry D., "Adaptive Prediction in Speech Differential Encoding Systems," Proceedings of the IEEE, Volume 68, Number 4, April 1980, pp. 488-525.
16. Golding, L.S., and Schultheiss, P.M., "Study of an Adaptive Quantizer," Proceedings of IEEE, Volume 55, Number 3, March 1967, pp. 293-297.
17. Goodman, David J., and Gersho, Allen, "Theory of an Adaptive Quantizer," IEEE Transaction on Communications, Volume COM-22, Number 8, August 1974, pp. 1037-1045.
18. Gray, Robert M., "Mutual Information Rate, Distortion, and Quantization in Metric Spaces," IEEE Transactions on Information Theory, Volume IT-26, Number 4, July 1980, pp. 412-422.
19. Habibi, Ali, "Hybrid Coding of Pictorial Data," IEEE Transactions on Communications, Volume COM-22, May 1974, pp. 614-624.
20. Huang, J.Y., and Schultheiss, P.M., "Block Quantization of Correlated Gaussian Random Variables," IEEE Transactions on Communication Systems, Volume CS-11, September 1963, pp. 289-296.
21. Jones, Richard A., Adaptive Hybrid Picture Coding, Technical Proposal D180-19071-1, Boeing Aerospace Company, Research & Engineering Divisions, September 1976.

22. Juang, B., Wong, D. Y., and Gray Jr., A. H., "Recent Developments in Vector Quantization for Speech Processing," Proceedings Int. Conf. of ASSP, April 1981.
23. Linde, Y., Buzo, A., and Gray, R. M., "An Algorithm for Vector Quantizer Design," IEEE Transactions on Communications, Volume COM-28, Number 1, January 1980, pp. 84-95.
24. Lloyd, S. P., "Least Squares Quantization in PCM," IEEE Transactions on Information Theory, Volume IT-28, Number 2, March 1982, pp. 129-137.
25. Max, Joel, "Quantizing for Minimum Distortion," IRE Transactions on Information Theory, Volume IT-6, March 1960, pp. 7-12.
26. Meditch, J. S., Stochastic Optimal Linear Estimation and Control, McGraw Hill Inc., New York, NY, 1969.
27. Melsa, James L., and Cohn, David L., Decision and Estimation Theory, McGraw Hill Inc., New York, NY, 1978.
28. Mitra, D., "Mathematical Analysis of an Adaptive Quantizer," Bell System Technical Journal, Volume 53, Number 5, May 1974, pp. 867-989.
29. Neagoe, V.E., "Predictive Data Compression of Color Picture Signals Using a Component Coding Method," International Communication Conference Record, Volume 2, Session 22.6, 1981.
30. Paez, M. D., and Glisson, T. H., "Minimum Mean-Squared-Error Quantization in Speech PCM and DPCM Systems," IEEE Transactions on Communications, Volume COM-20, April 1972, pp. 225-230.
31. Pearlman, W. A., and Senge, G. H., "Optimal Quantization of the Rayleigh Probability Distribution," IEEE Transactions on Communications, Volume COM-27, Number 1, January 1979, pp. 101-112.
32. Pratt, William K., Digital Image Processing, John-Wiley & Sons Inc., New York, NY, 1978.
33. Roe, Glenn M., "Quantizing for Minimum Distortion," IEEE Transactions on Information Theory, Volume IT-10, October 1964, pp. 384-385.

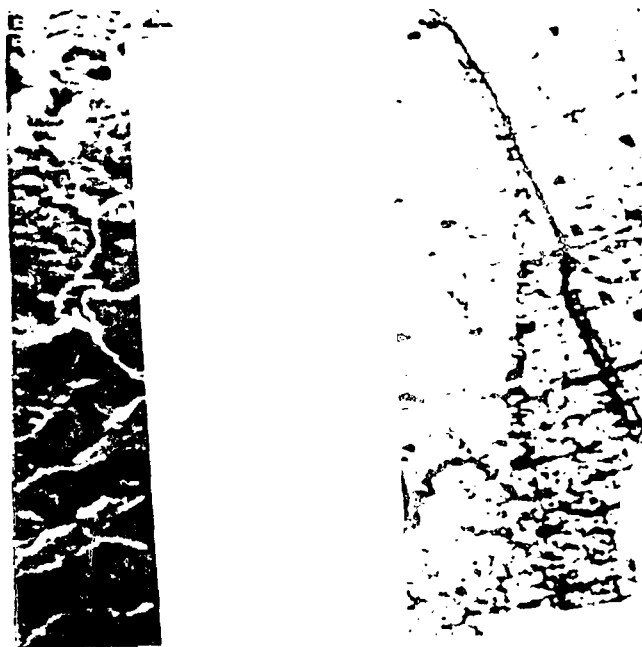
34. Slepian, David, "Permutation Modulation,"
Proceedings of the IEEE, Volume 53, March 1965,
pp. 228-236.
35. Stratigakis, Alexios V., Adaptive Stochastic
Predictive Coding, unpublished M.S.E.E. thesis,
University of Arkansas, 1983.
36. Wyner, Aaron D., "Fundamental Limits in
Information Theory," Proceedings of IEEE,
Volume 69, Number 2, February 1981 pp. 239-251.
37. Zador, P. L., "Asymptotic Quantization Error of
Continuous Signals and the Quantization
Dimension," IEEE Transactions on Information
Theory, Volume IT-28, Number 2, March 1982, pp.
139-149.
38. Zetterberg, Lars, Ericsson, Staffan, and
Brusewitz, Harald, "Interframe DPCM with
Adaptive Quantization and Entropy Coding," IEEE
Transactions on Communications, Volume COM-30,
Number 8, August 1982, PP. 1888-1899.

Comments On Photos and Histograms

- 1) Error Histograms of each image appear on the page immediately following the photograph.
- 2) Variations in vector quantizer simulations (VAHPC) data rates occur due to variations in the partial code book transmission distortion threshold.
- 3) Note the blurring of the left edge in the backward adaptive quantizing scheme photographs. This is due to the fact that step size estimation is a real-time past-value situation and a number of samples must be passed before any accuracy can occur. This may be remedied by further investigation as to optimum starting value estimates, and or modification of the step size estimation algorithm.
- 4) The backward adaptive simulation was run with $\alpha_1=0.7$ and $\alpha_2=1.0$.
- 5) The backward adaptive scheme failed almost completely for any residual quantizer using only one-bit resolution. This is due to the fact that the step size estimate is obtained from quantized residual values. Hence, the photographs and error histograms of this simulation was not included.



Original MIT monochrome GIRL image (128 x 128)



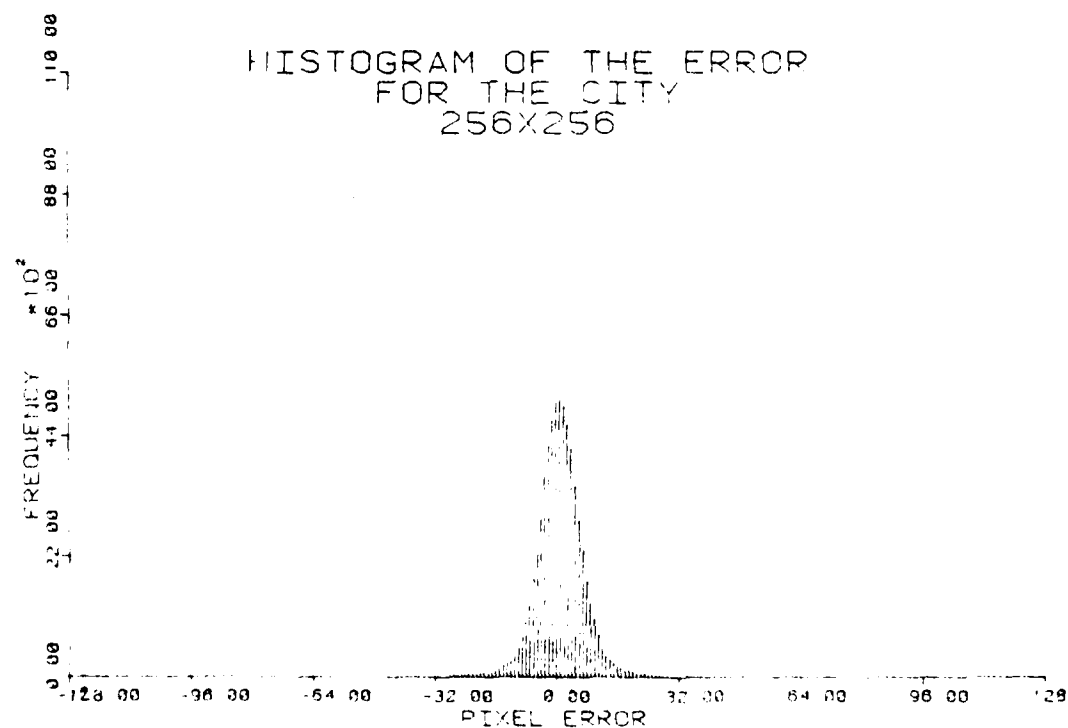
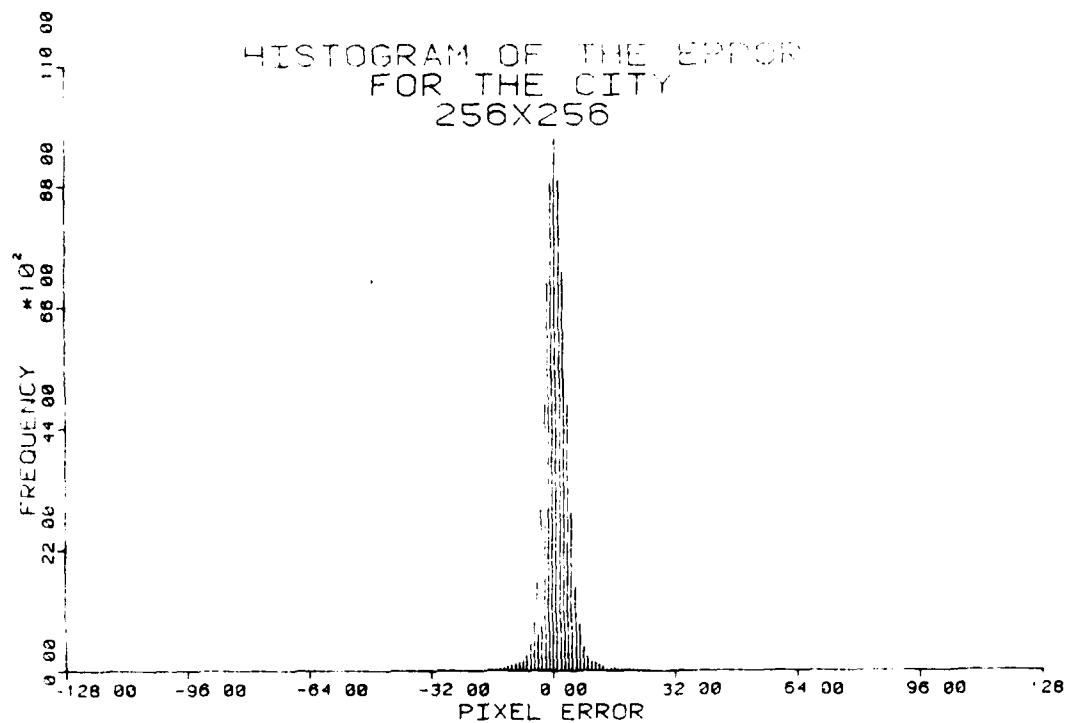
Original aerial photo of the CITY image (256 x 256)

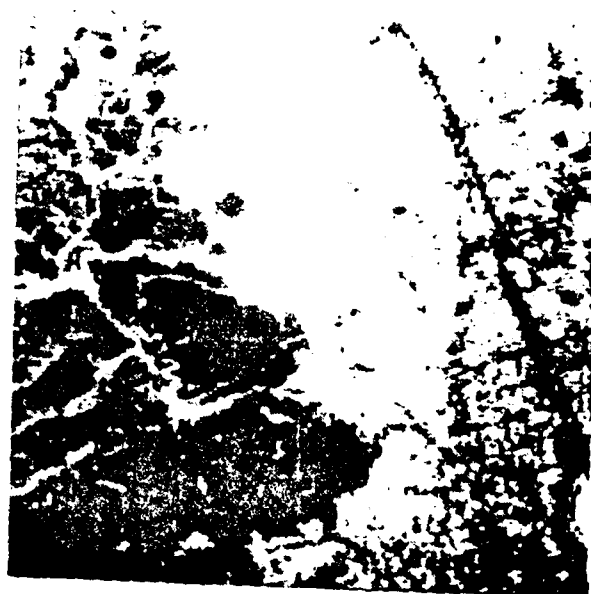


ASPC / ARMA model / Forward Adaptive Max
Data rate = 3.15 bits/pel SNR = 20.78 dB

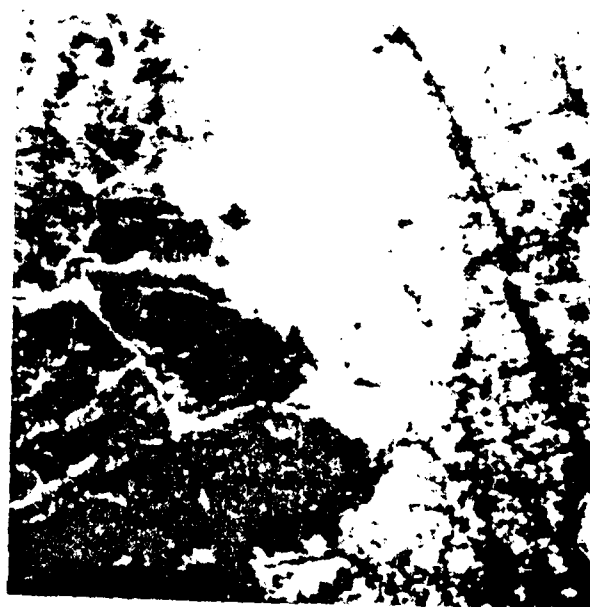


ASPC / ARMA model / Forward Adaptive Max
Data rate = 2.15 bits/pel SNR = 15.77 dB

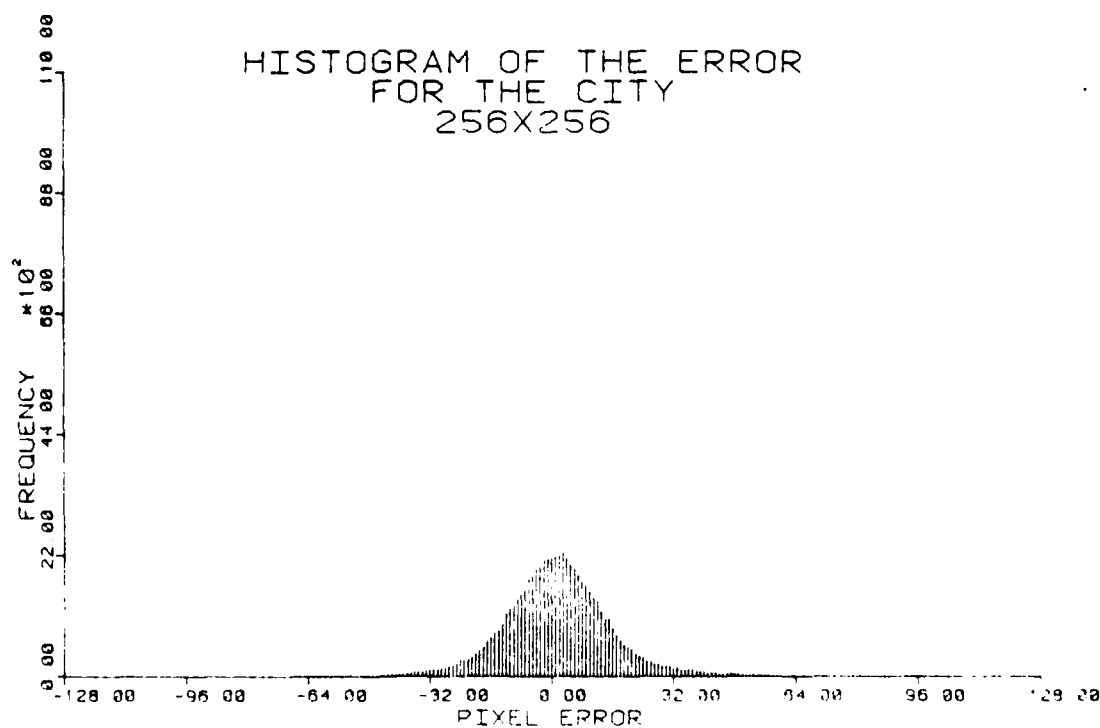
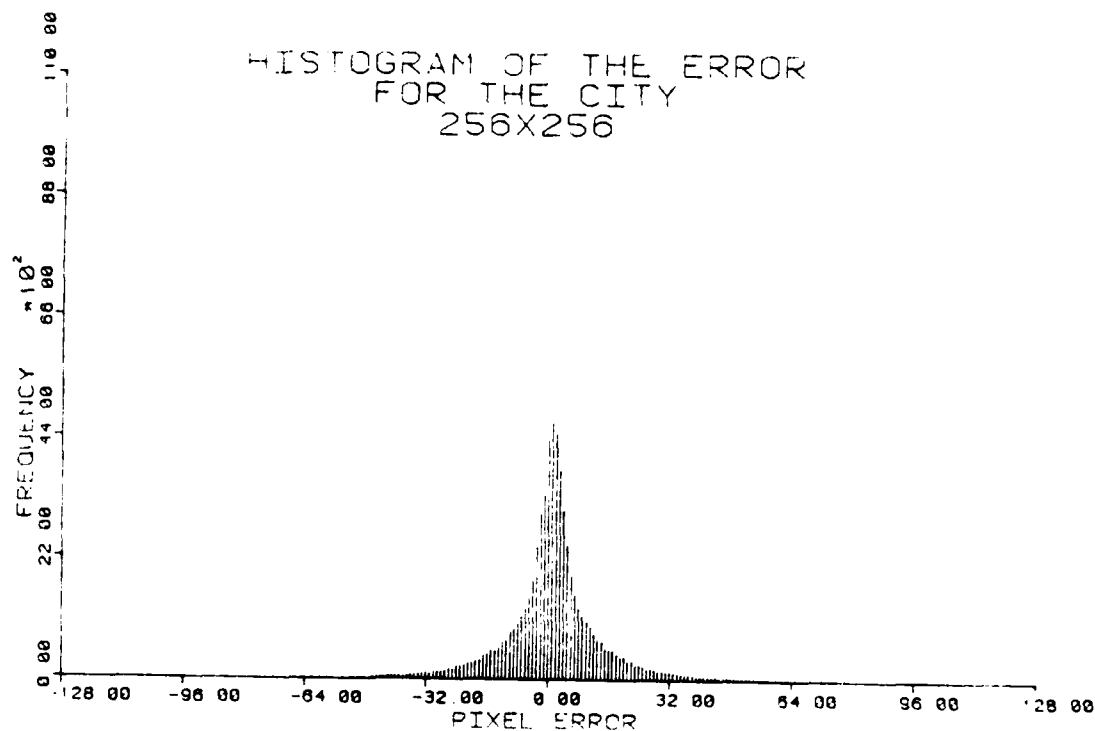


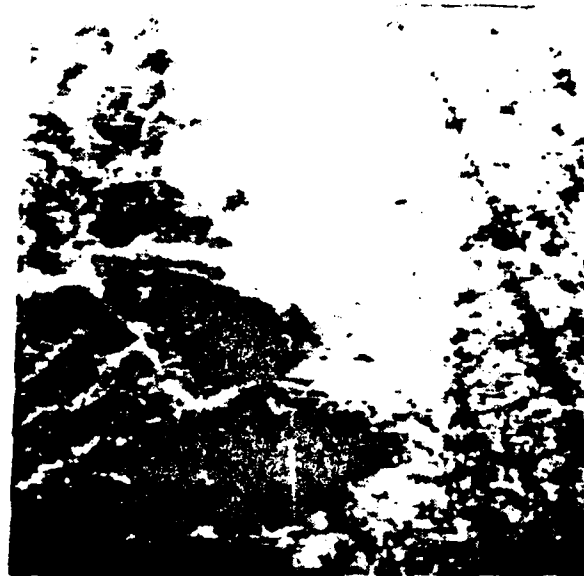


ASPC / ARMA model / Forward Adaptive Max
- ZEROING -
Data rate = 1.65 bits/pel SNR = 10.22 dB

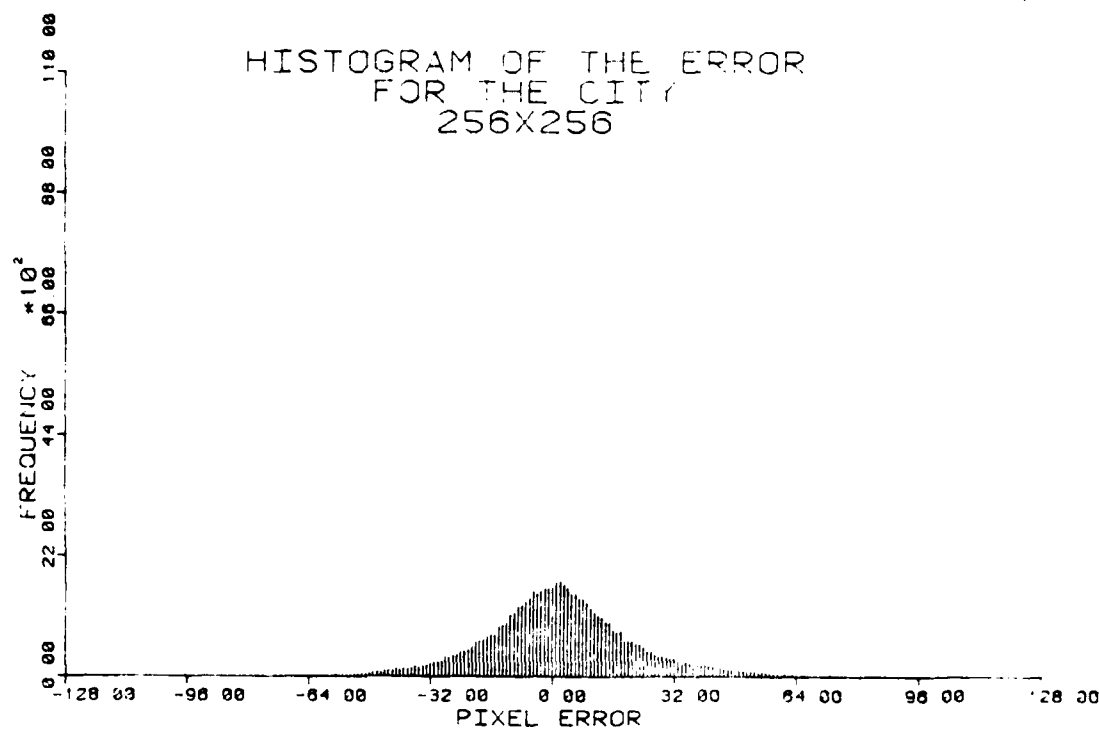


ASPC / ARMA model / Forward Adaptive Max
Data rate = 1.15 bits/pel SNR = 10.02 dB

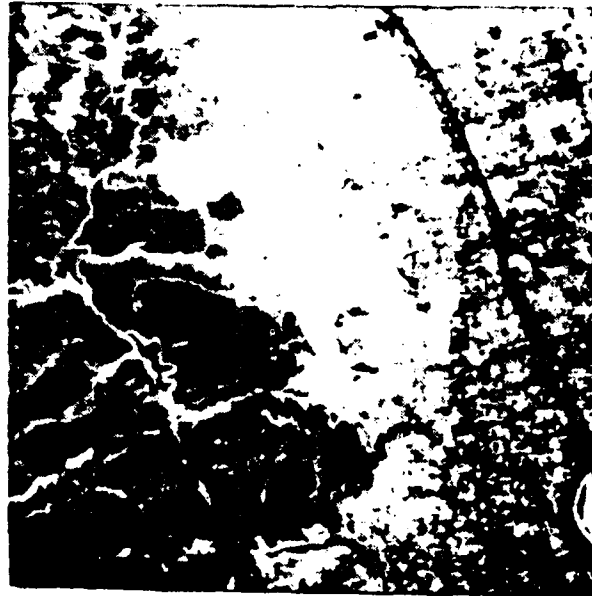




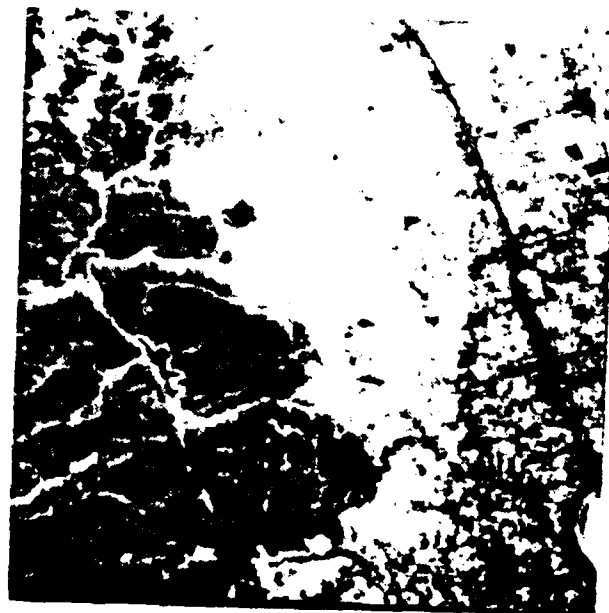
ASPC / ARMA model / Forward Adaptive Max
- ZEROING -
Data rate = 0.65 bits/pel SNR = 6.30 dB



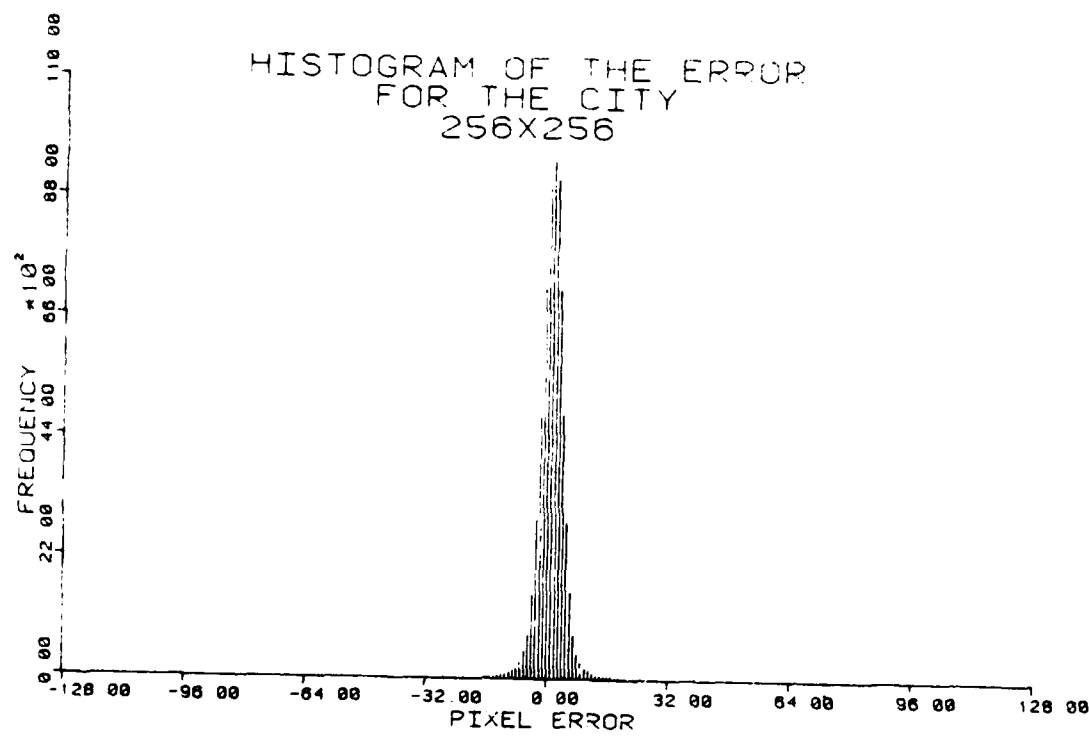
ASPC / ARMA model / Forward Adaptive Max
- ZEROING -
Data rate = 0.65 bits/pel SNR = 6.30 dB



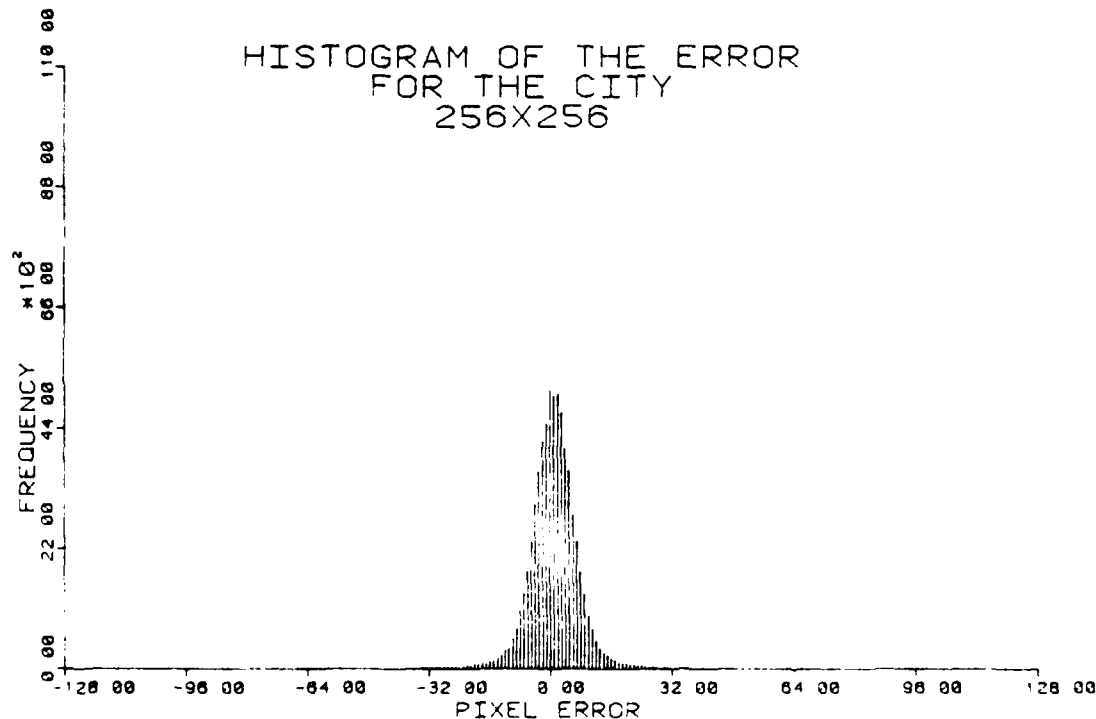
ASPC / Kalman filter / Forward Adaptive Max
Data rate = 3.15 bits/pel SNR = 20.84 dB



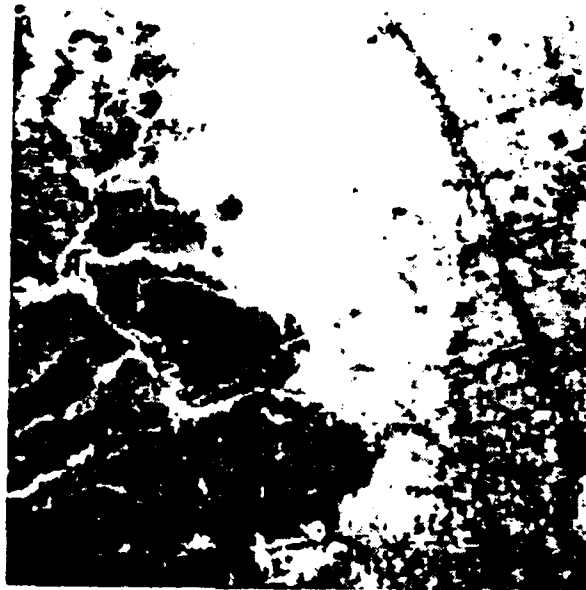
ASPC / Kalman filter / Forward Adaptive Max
Data rate = 2.15 bits/pel SNR = 15.74 dB



ASPC / Kalman filter / Forward Adaptive Max
Data rate = 3.15 bits/pel SNR = 20.84 dB



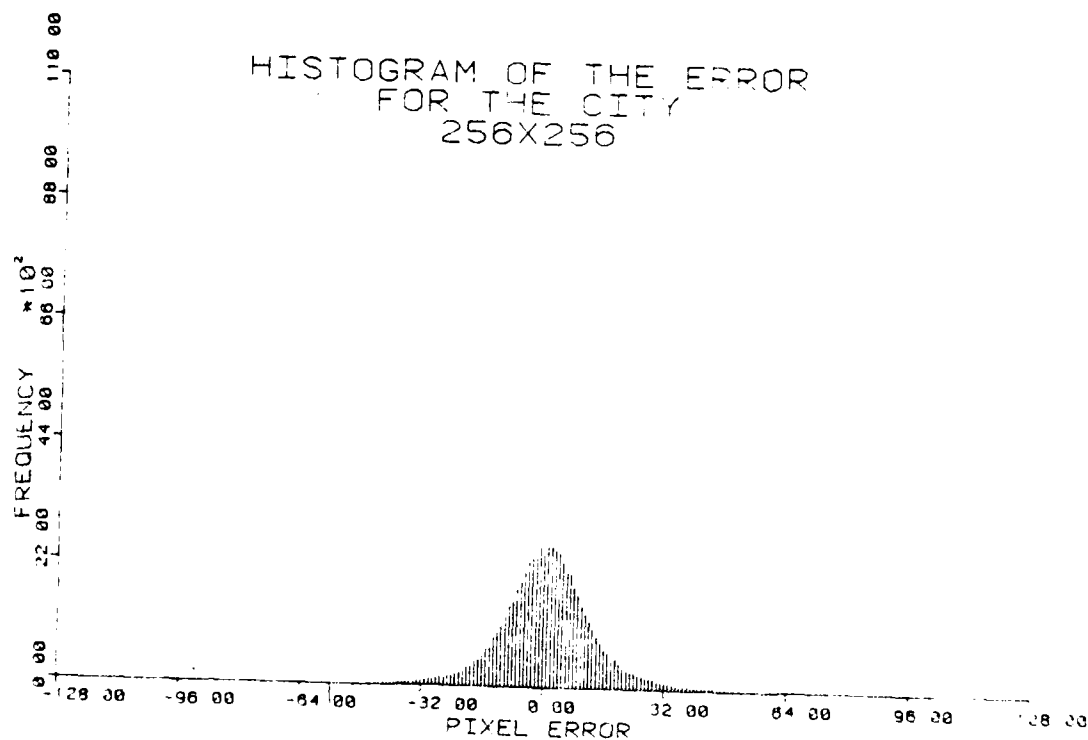
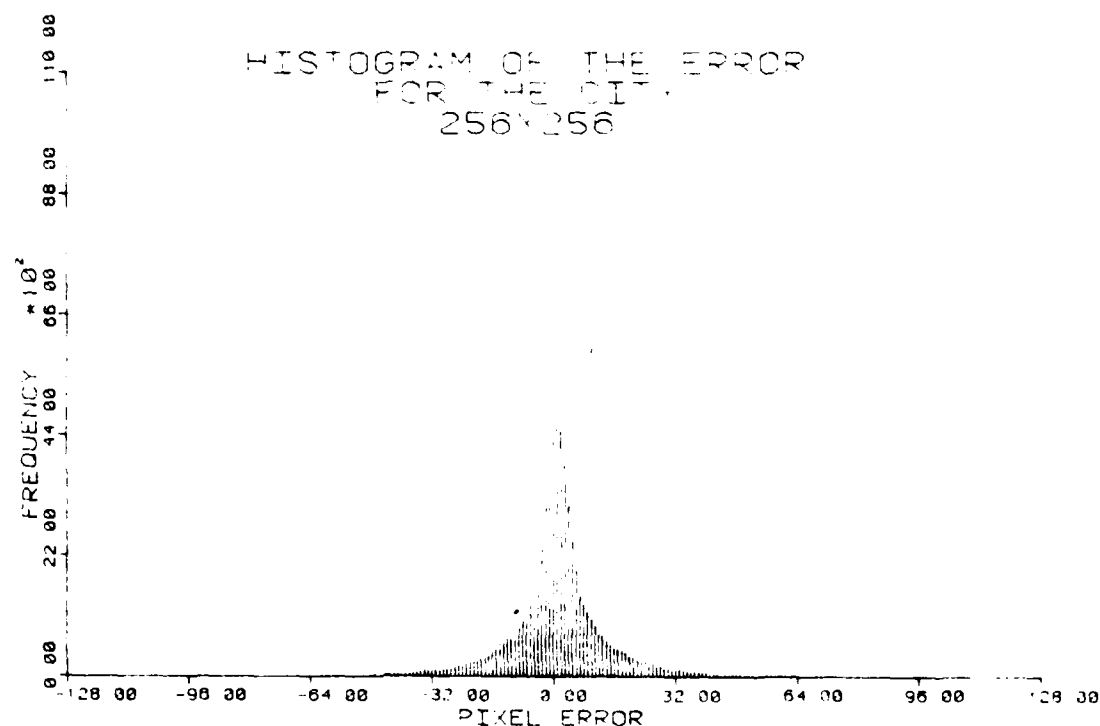
ASPC / Kalman filter / Forward Adaptive Max
Data rate = 2.15 bits/pel SNR = 15.74 dB

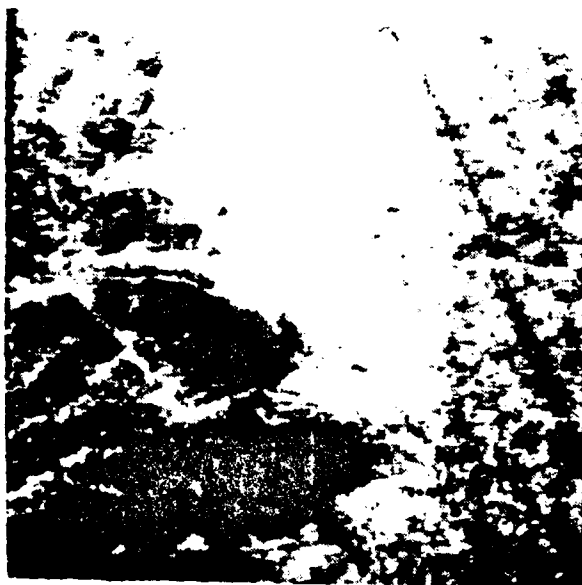


ASPC / Kalman filter / Forward Adaptive Max
- ZEROING -
Data rate = 1.65 bits/pel SNR = 10.26 dB

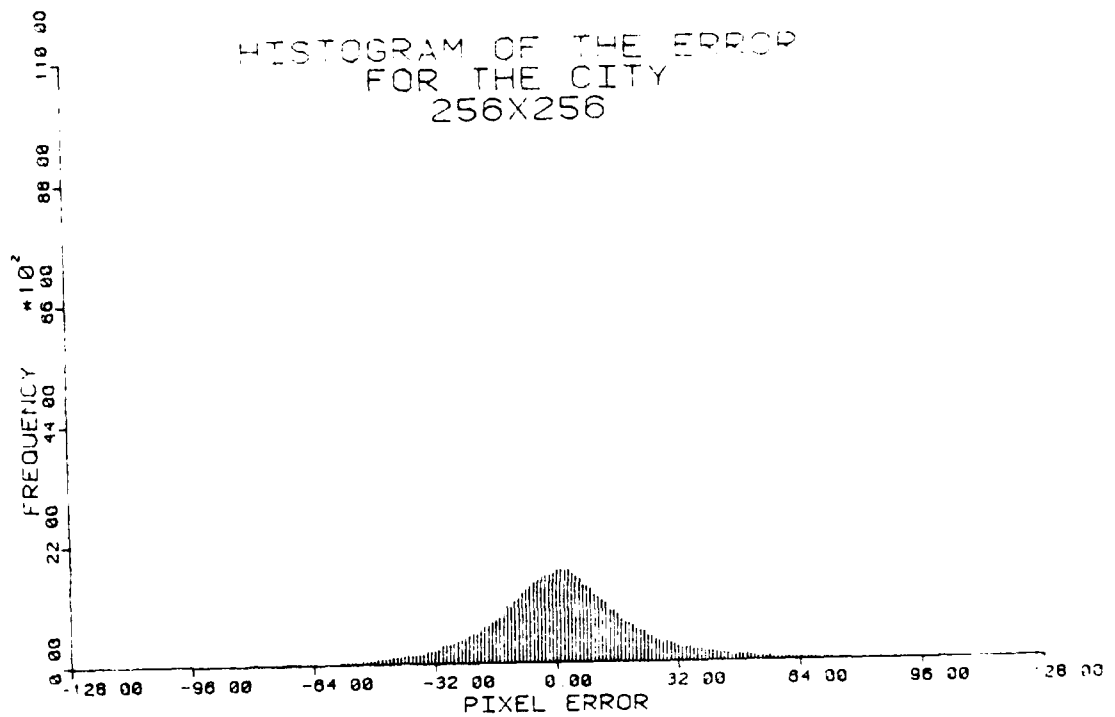


ASPC / Kalman filter / Forward Adaptive Max
Data rate = 1.15 bits/pel SNR = 10.11 dB





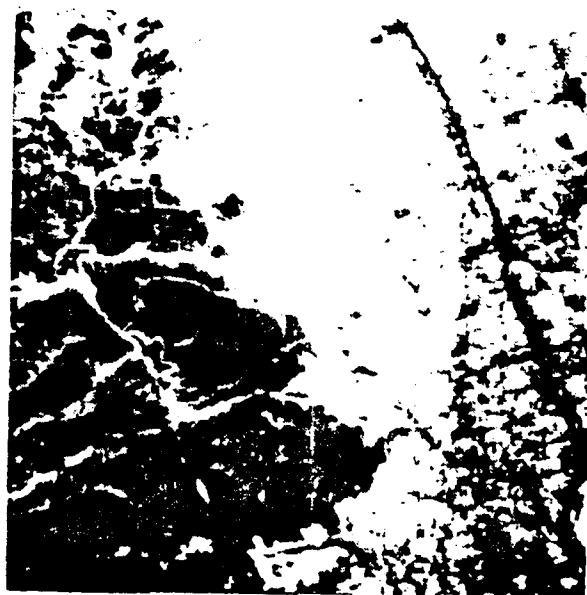
ASPC / Kalman filter / Forward Adaptive Max
- ZEROING -
Data rate = 0.65 bits/pel SNR = 6.40 dB



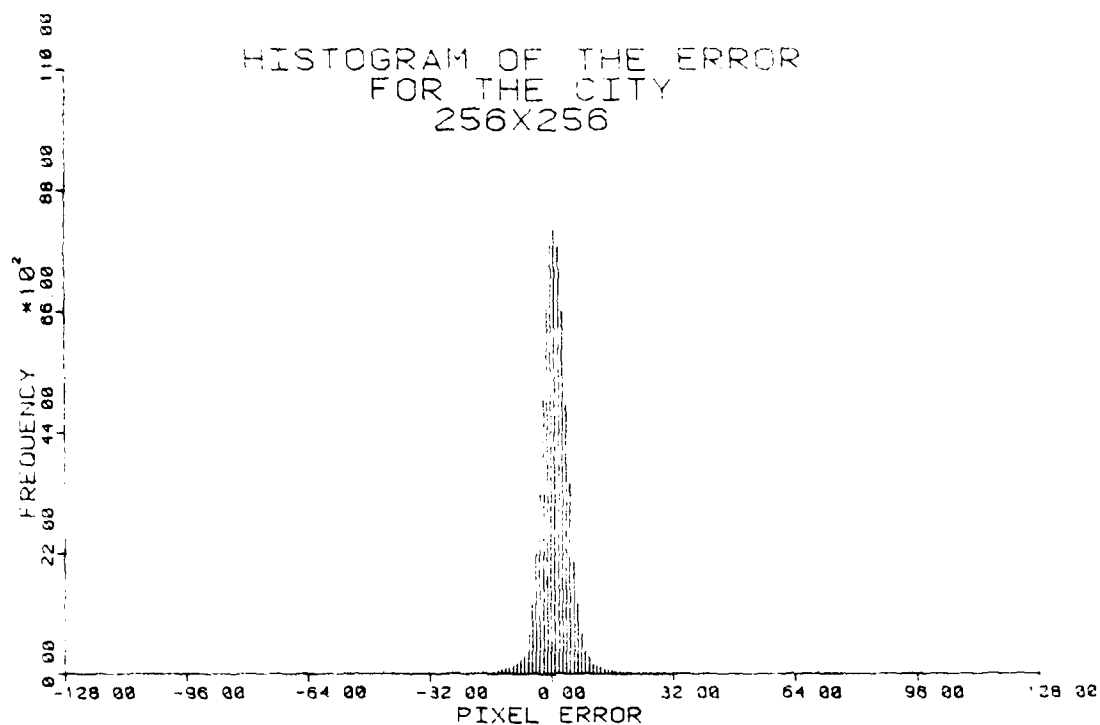
ASPC / Kalman filter / Forward Adaptive Max
- ZEROING -
Data rate = 0.65 bits/pel SNR = 6.40 dB



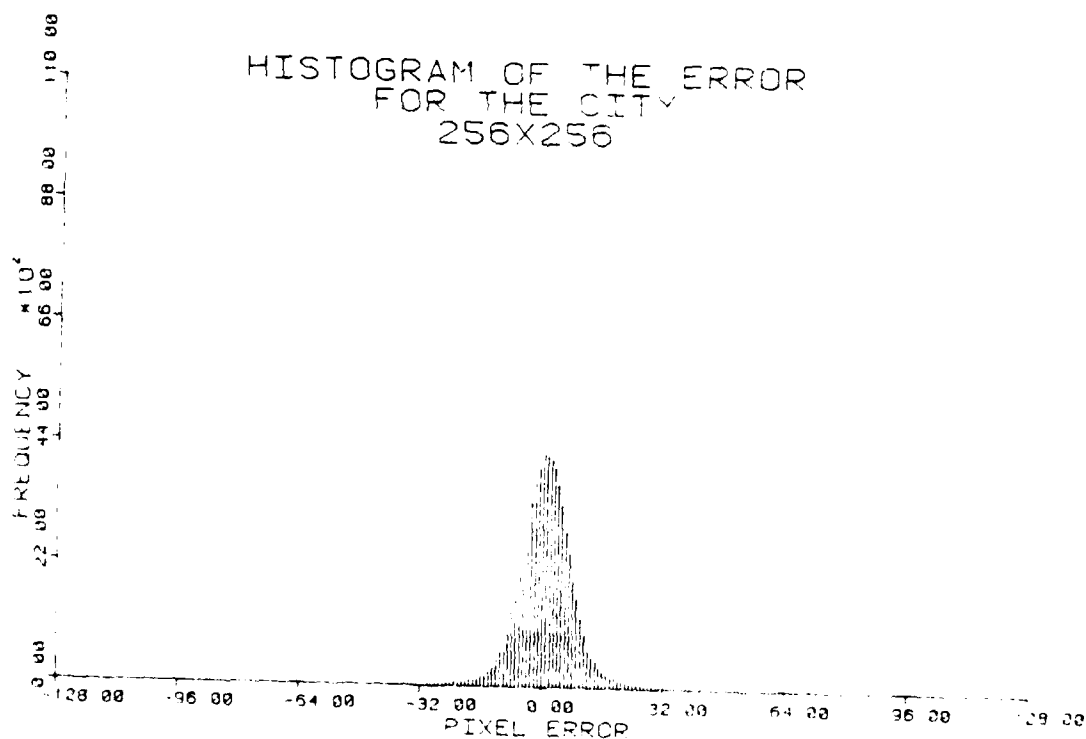
ASPC / Stoch. Approx. / Forward Adaptive Max
Data rate = 3.15 bits/pel SNR = 18.88 dB



ASPC / Stoch. Approx. / Forward Adaptive Max
Data rate = 2.15 bits/pel SNR = 14.04 dB



ASPC / Stoch. Approx. / Forward Adaptive Max
Data rate = 3.15 bits/pel SNR = 18.88 dB



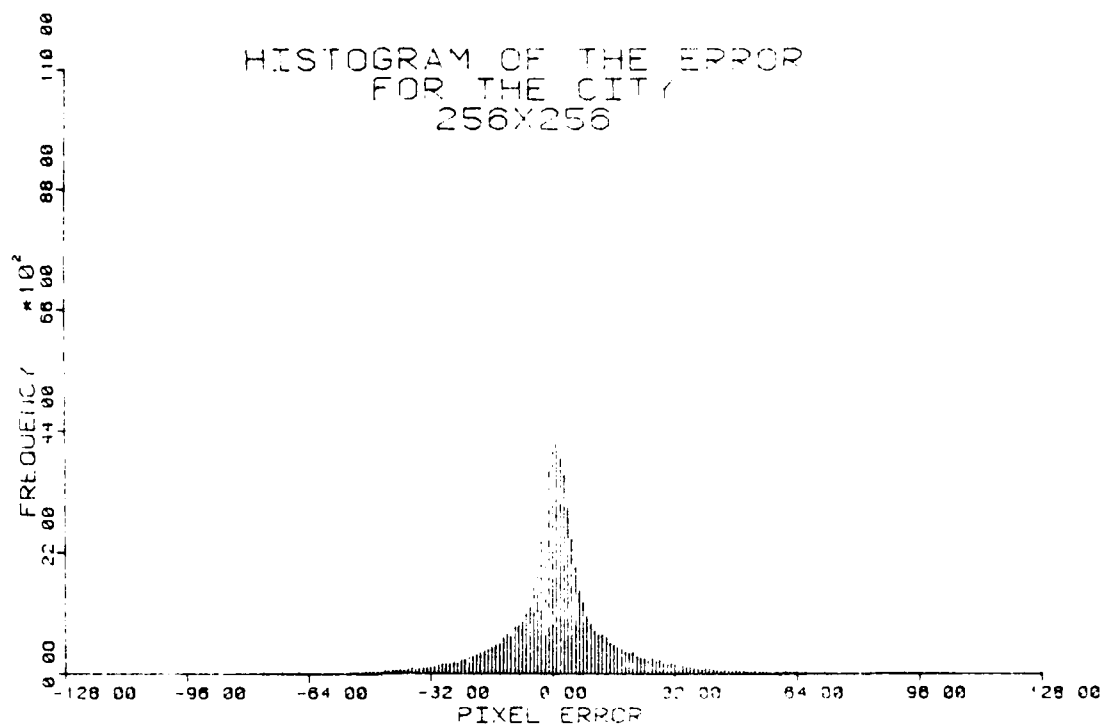
ASPC / Stoch. Approx. / Forward Adaptive Max
Data rate = 2.15 bits/pel SNR = 14.04 dB



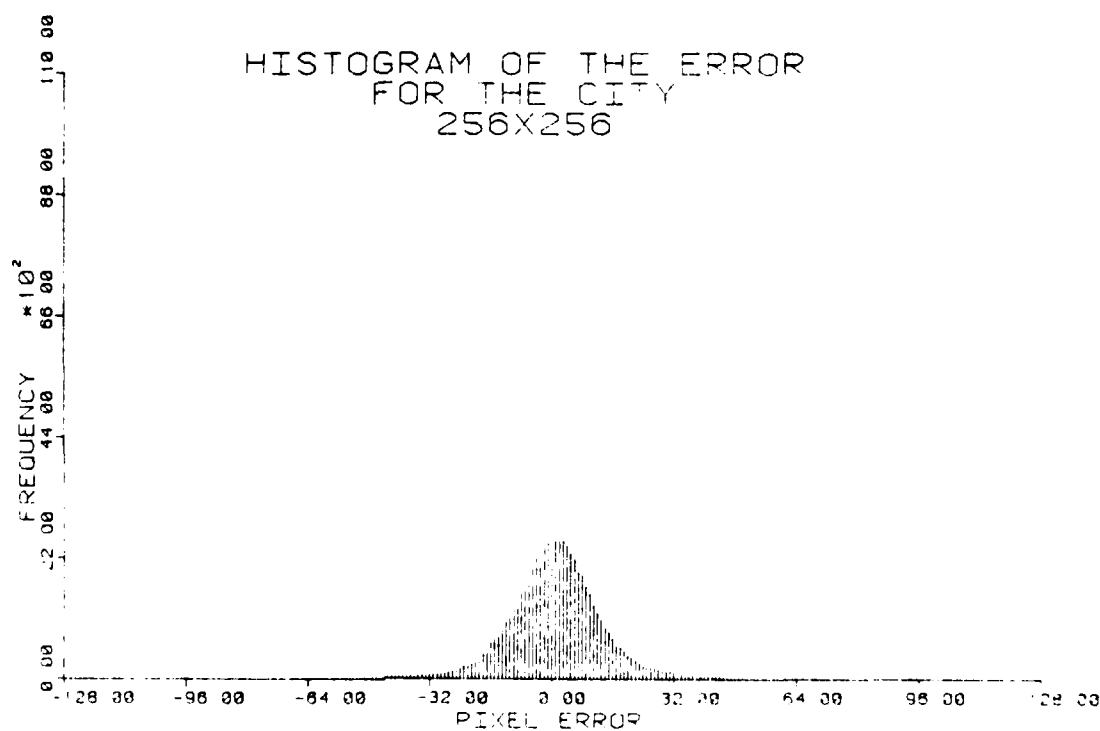
ASPC / Stoch. Approx. / Forward Adaptive Max
- ZEROING -
Data rate = 1.65 bits/pel SNR = 8.83 dB



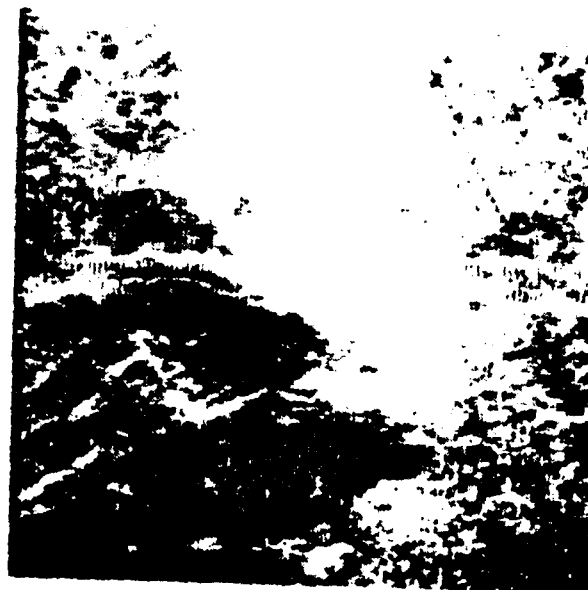
ASPC / Stoch. Approx. / Forward Adaptive Max
Data rate = 1.15 bits/pel SNR = 8.60 dB



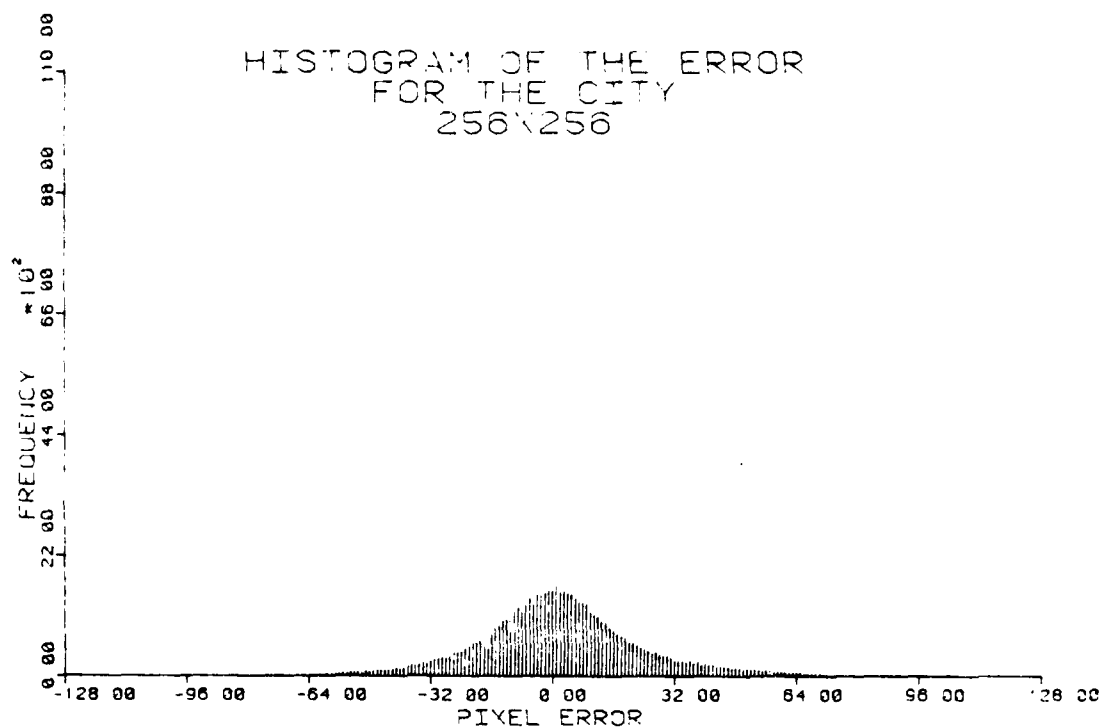
ASPC / Stoch. Approx. / Forward Adaptive Max
- ZEROING -
Data rate = 1.65 bits/pel SNR = 8.83 dB



ASPC / Stoch. Approx. / Forward Adaptive Max
Data rate = 1.15 bits/pel SNR = 8.60 dB



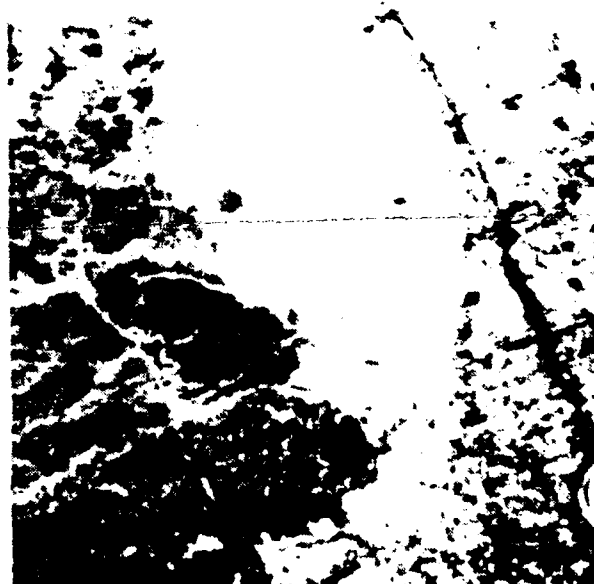
ASPC / Stoch. Approx. / Forward Adaptive Max
- ZEROING -
Data rate = 0.65 bits/pel SNR = 5.50 dB



ASPC / Stoch. Approx. / Forward Adaptive Max
- ZEROING -
Data rate = 0.65 bits/pel SNR = 5.50 dB

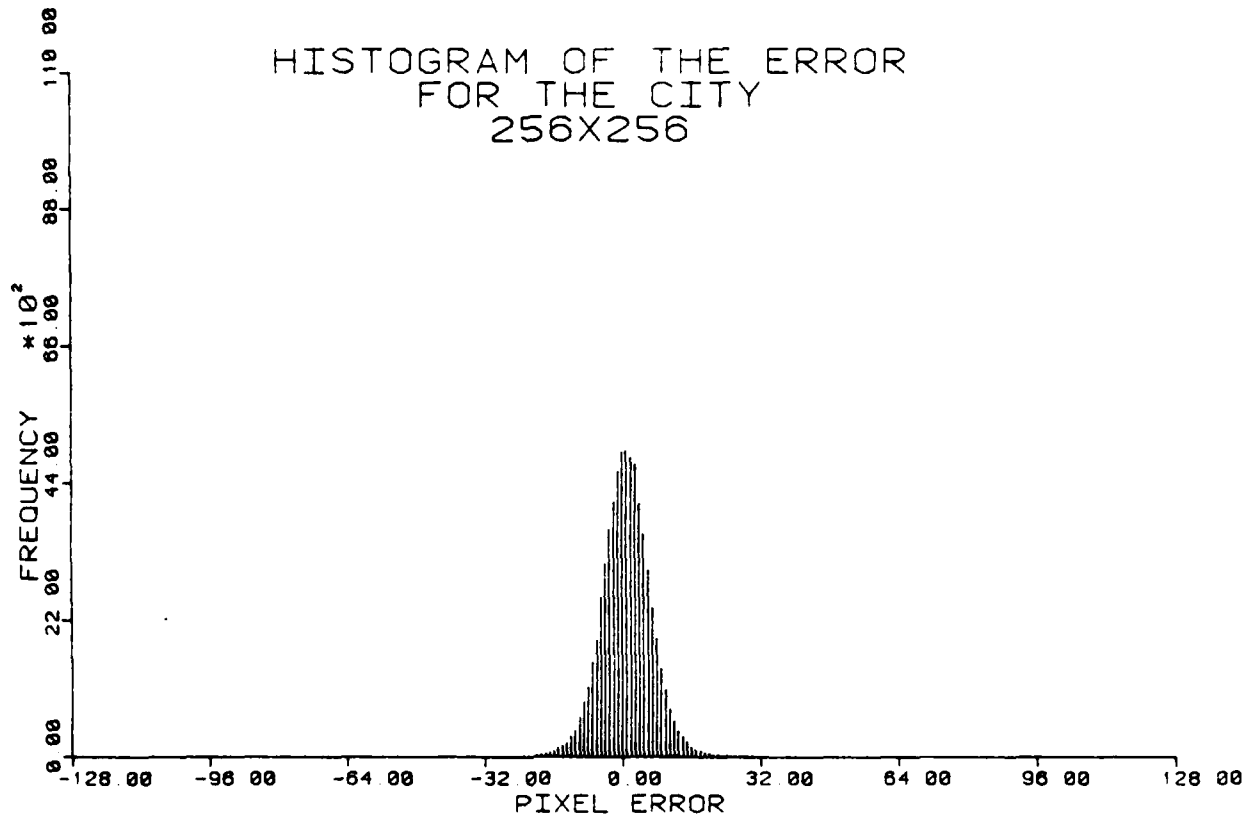


VAHPC / 16 Code Vectors / DIM = 2
Data rate = 2.35 bits/pel SNR = 17.12 dB



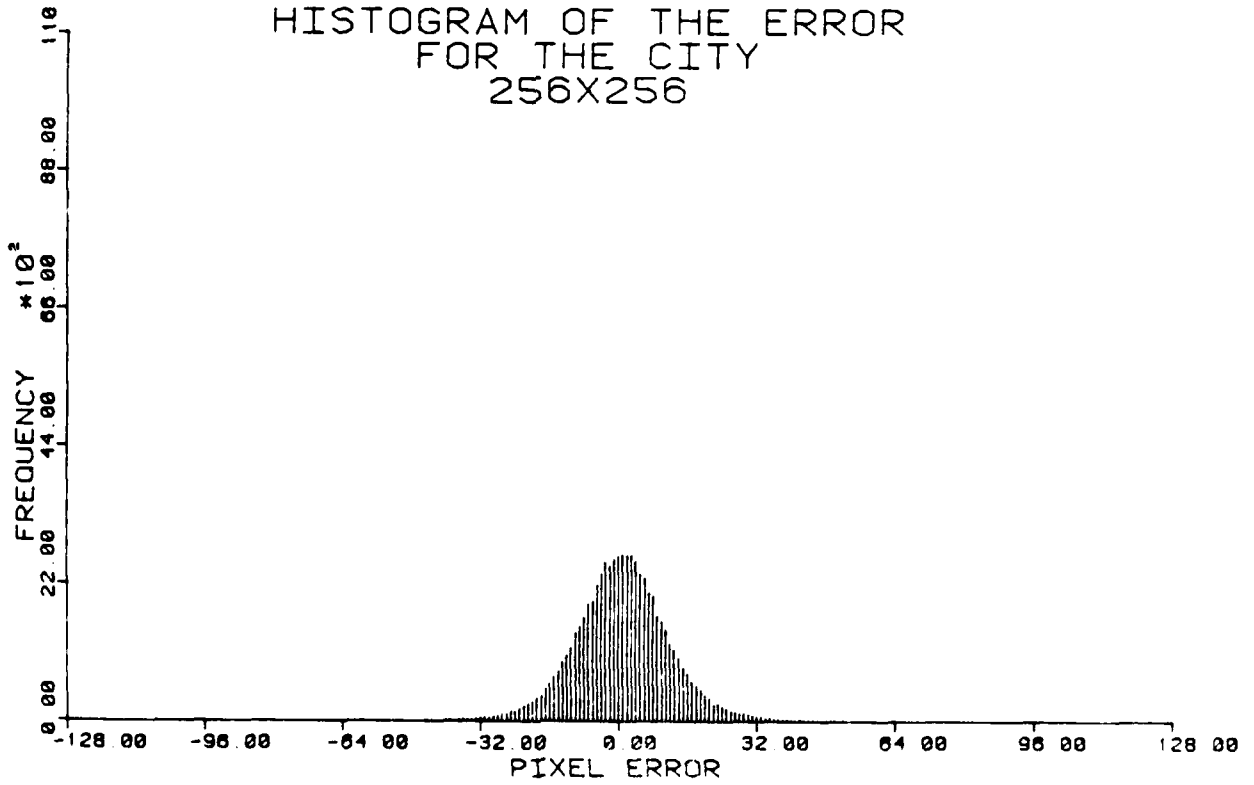
VAHPC / 16 Code Vectors / DIM = 4
Data rate = 1.45 bits/pel SNR = 11.44 dB

HISTOGRAM OF THE ERROR FOR THE CITY 256X256



VAHPC / 16 Code Vectors / DIM = 2
Data rate = 2.35 bits/pel SNR = 17.12 dB

HISTOGRAM OF THE ERROR FOR THE CITY 256X256



VAHPC / 16 Code Vectors / DIM = 4
Data rate = 1.45 bits/pel SNR = 11.44 dB
121

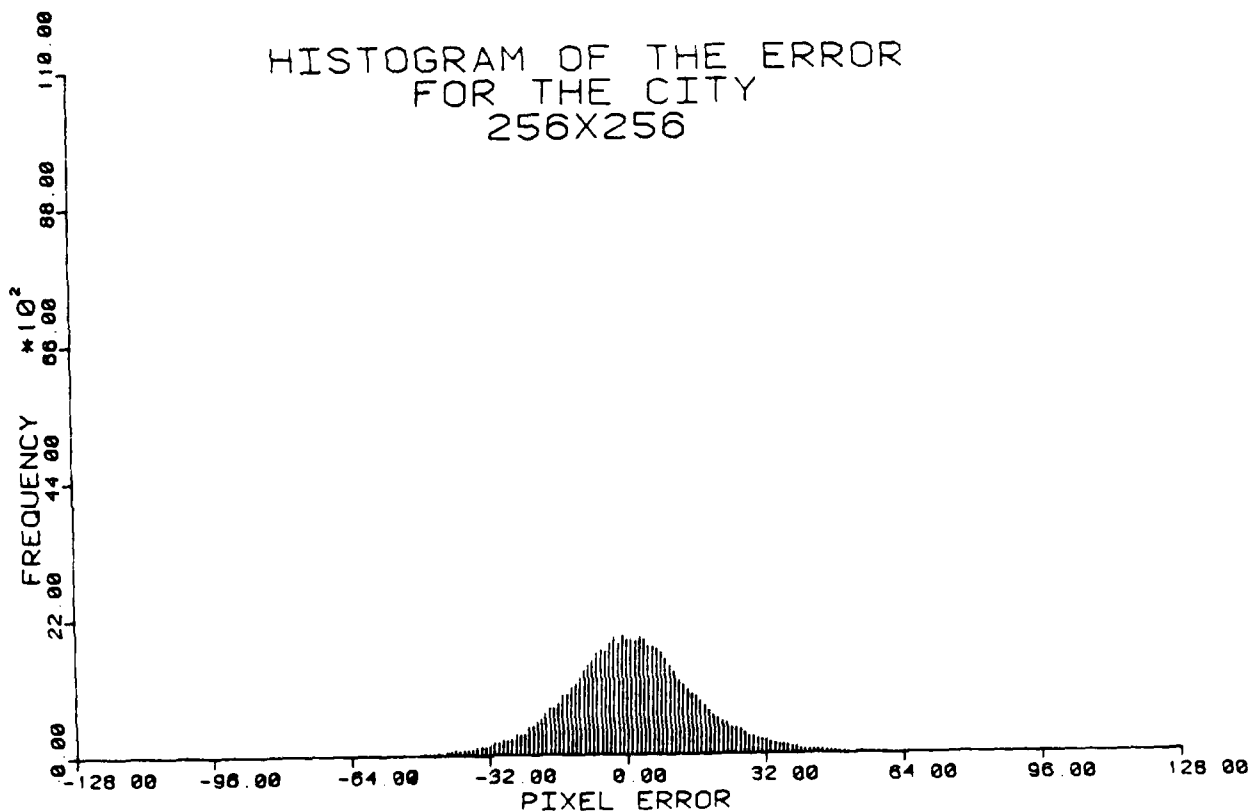


VAHPC / 16 Code Vectors / DIM = 16
Data rate = 0.84 bits/pel SNR = 8.58 dB



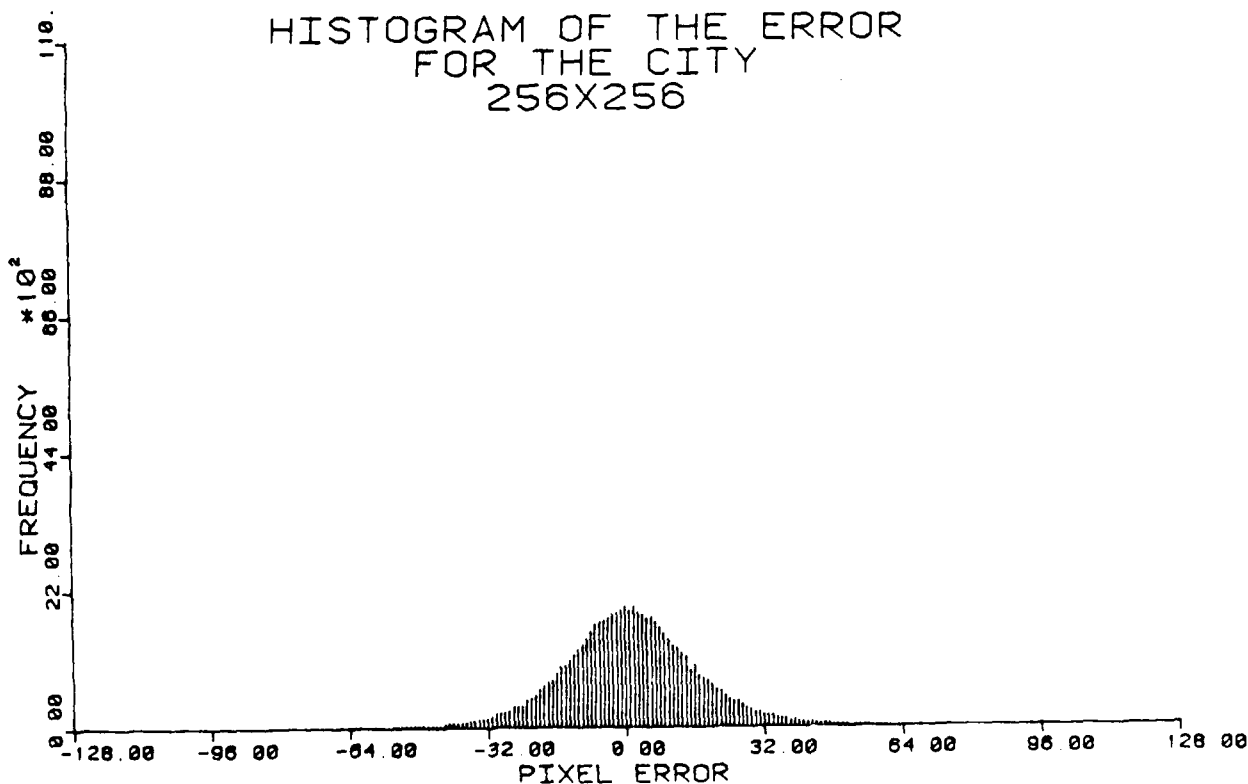
VAHPC / 16 Code Vectors / DIM = 16
Data rate = 0.83 bits/pel SNR = 8.57 dB

HISTOGRAM OF THE ERROR FOR THE CITY 256X256



VAHPC / 16 Code Vectors / DIM = 16
Data rate = 0.84 bits/pel SNR = 8.58 dB

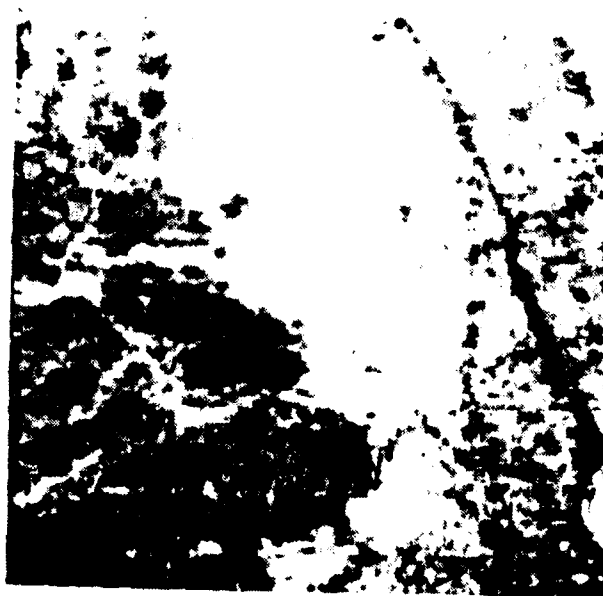
HISTOGRAM OF THE ERROR FOR THE CITY 256X256



VAHPC / 16 Code Vectors / DIM = 16
Data rate = 0.83 bits/pel SNR = 8.57 dB

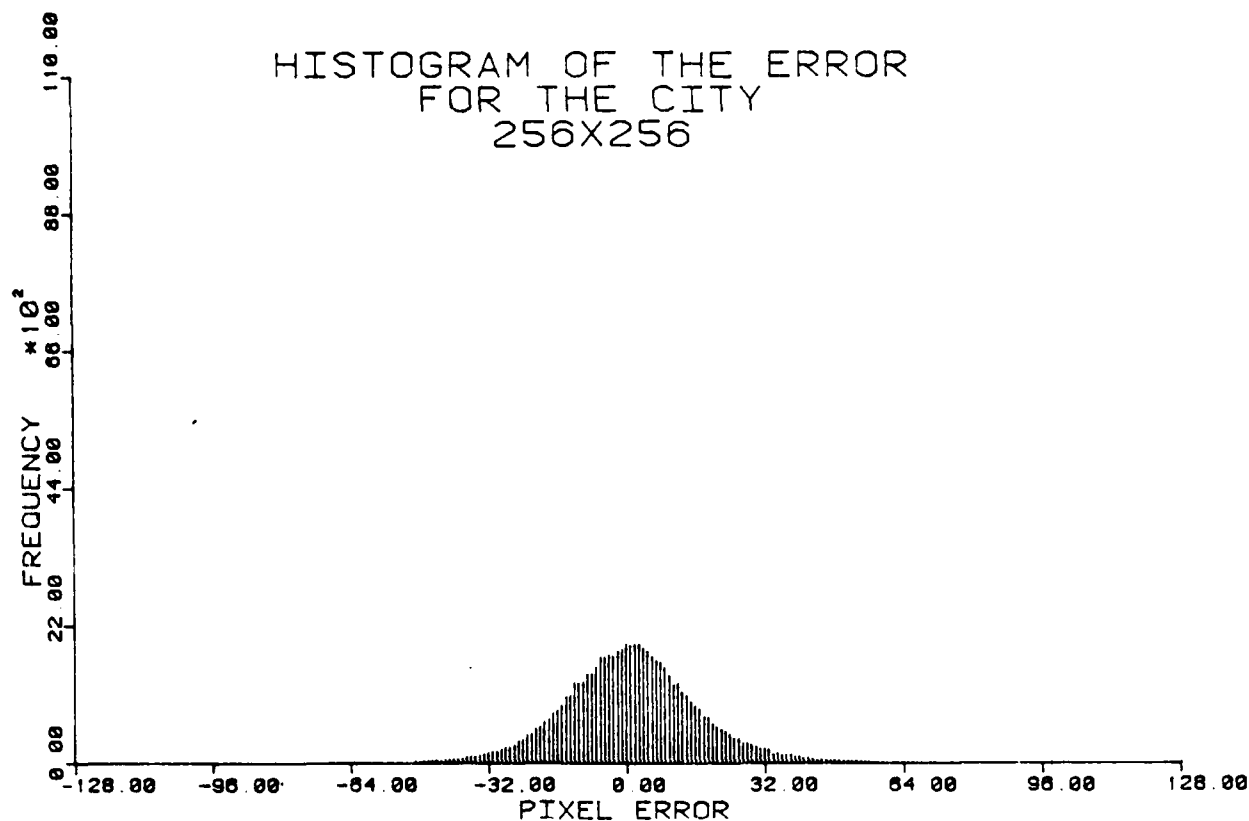


VAHPC / 16 Code Vectors / DIM = 16
Data rate = 0.77 bits/pel SNR = 8.43 dB



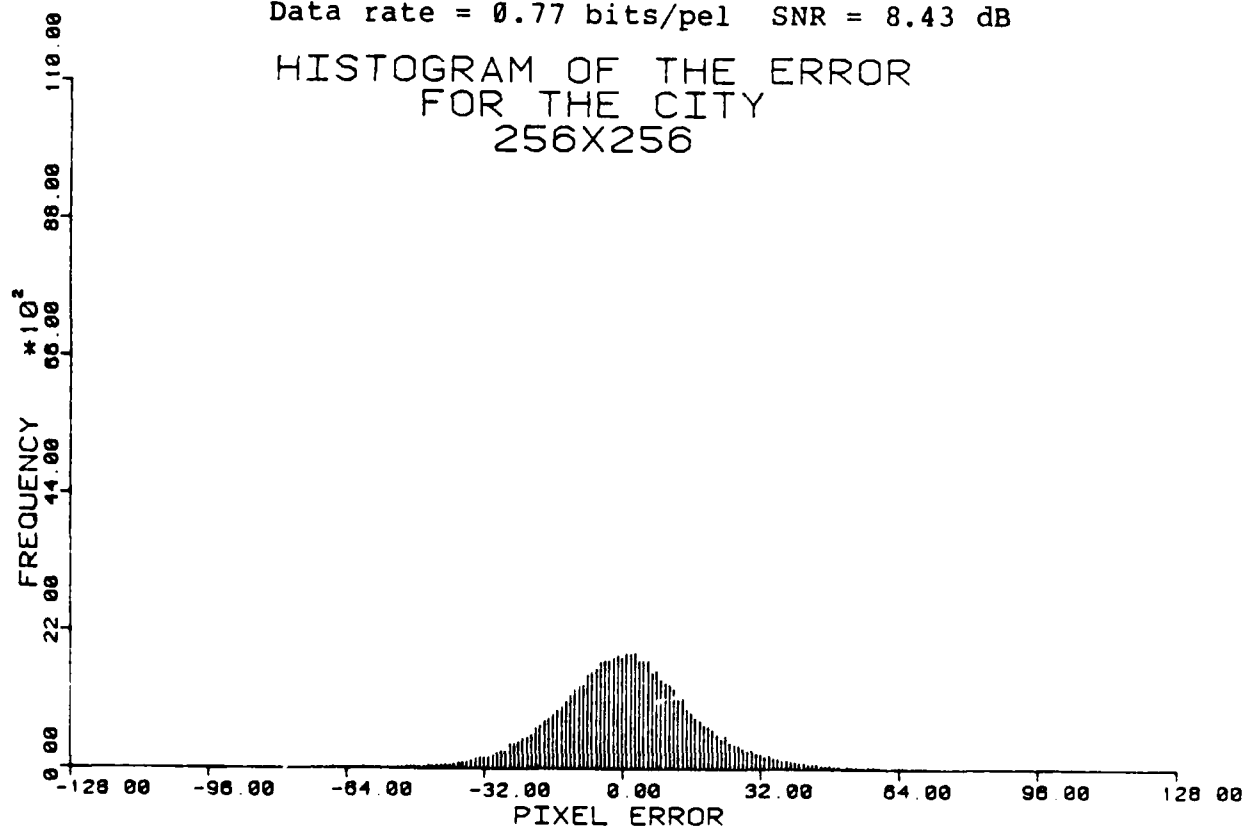
VAHPC / 16 Code Vectors / DIM = 16
Data rate = 0.67 bits/pel SNR = 8.20 dB

HISTOGRAM OF THE ERROR FOR THE CITY 256X256

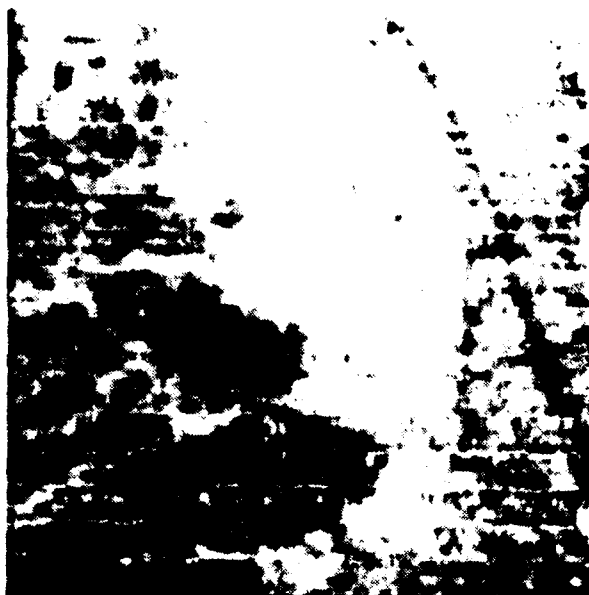


VAHPC / 16 Code Vectors / DIM = 16
Data rate = 0.77 bits/pel SNR = 8.43 dB

HISTOGRAM OF THE ERROR FOR THE CITY 256X256

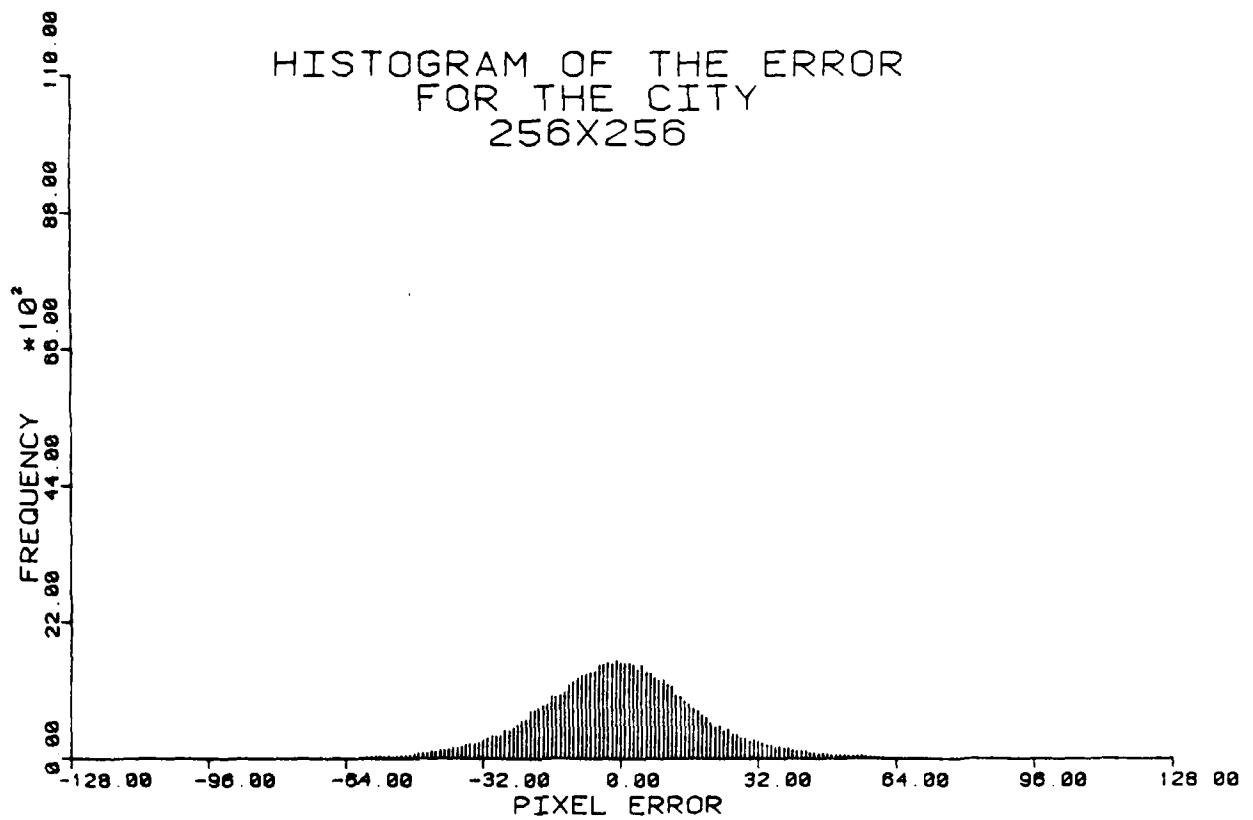


VAHPC / 16 Code Vectors / DIM = 16
Data rate = 0.67 bits/pel SNR = 8.20 dB

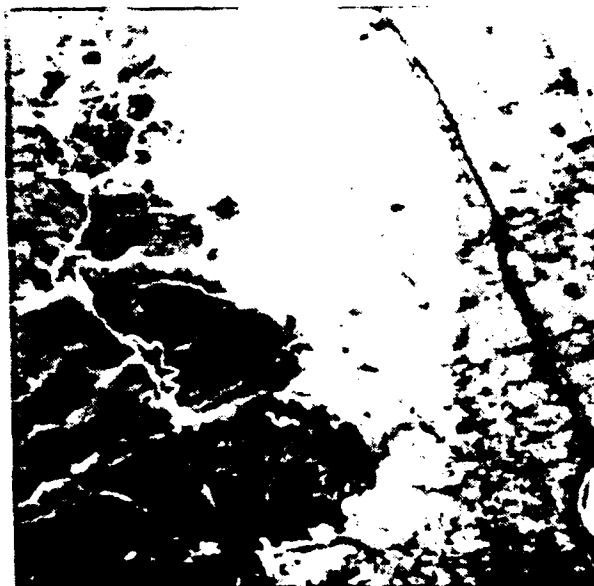


VAHPC / 8 Code Vectors / DIM = 16
Data rate = 0.28 bits/pel SNR = 6.72 dB

HISTOGRAM OF THE ERROR
FOR THE CITY
256X256



VAHPC / 8 Code Vectors / DIM = 16
Data rate = 0.28 bits/pel SNR = 6.72 dB

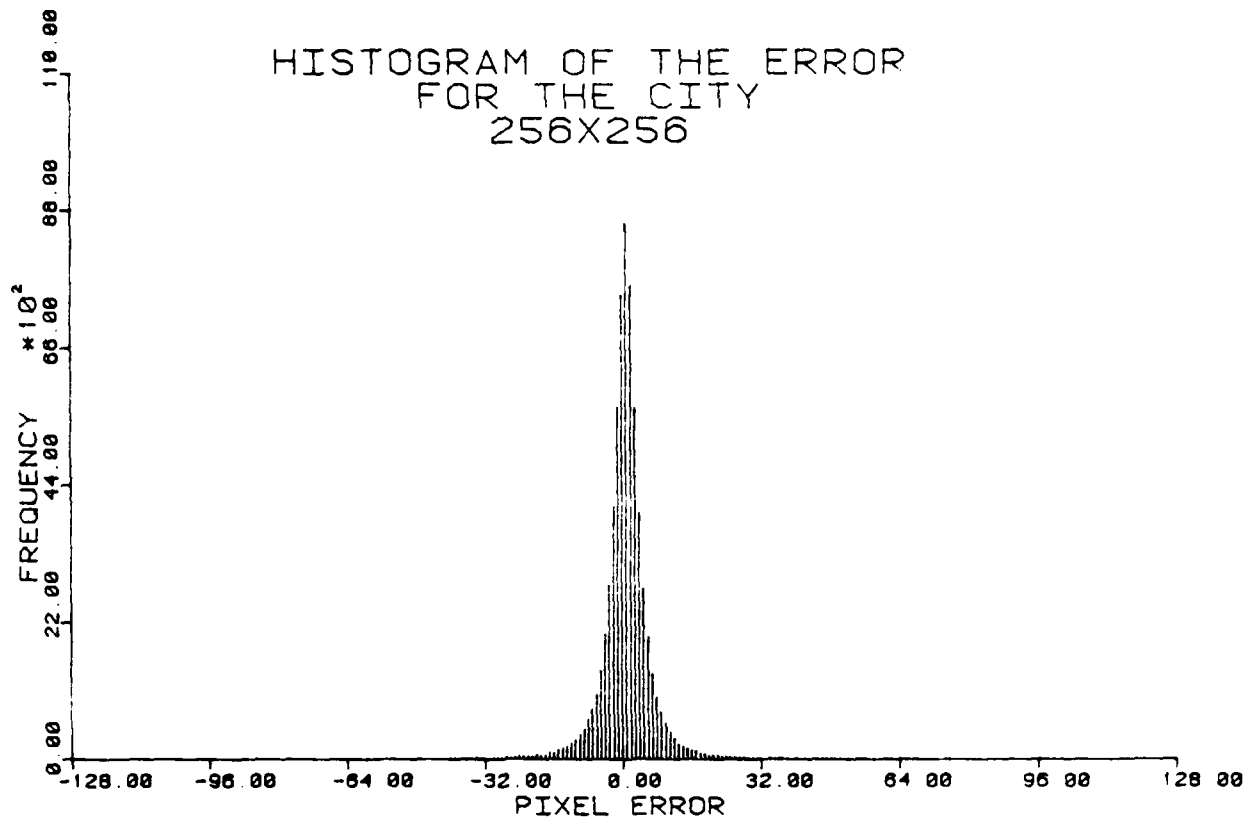


ASPC / Kalman filter / Backward Adaptive Max
Data rate = 3.12 bits/pel SNR = 16.11 dB



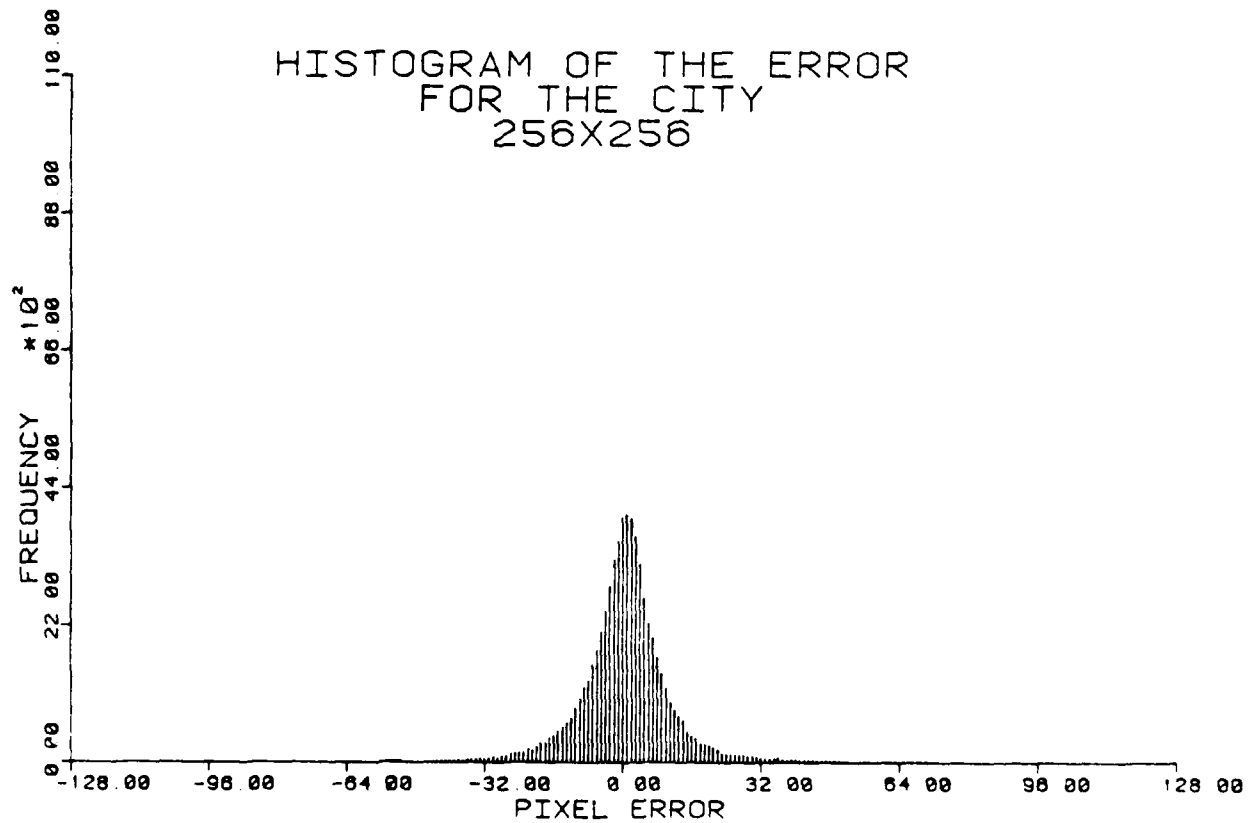
ASPC / Kalman filter / Backward Adaptive Max
Data rate = 2.12 bits/pel SNR = 11.47 dB

HISTOGRAM OF THE ERROR FOR THE CITY 256X256



ASPC / Kalman filter / Backward Adaptive Max
Data rate = 3.12 bits/pel SNR = 16.11 dB

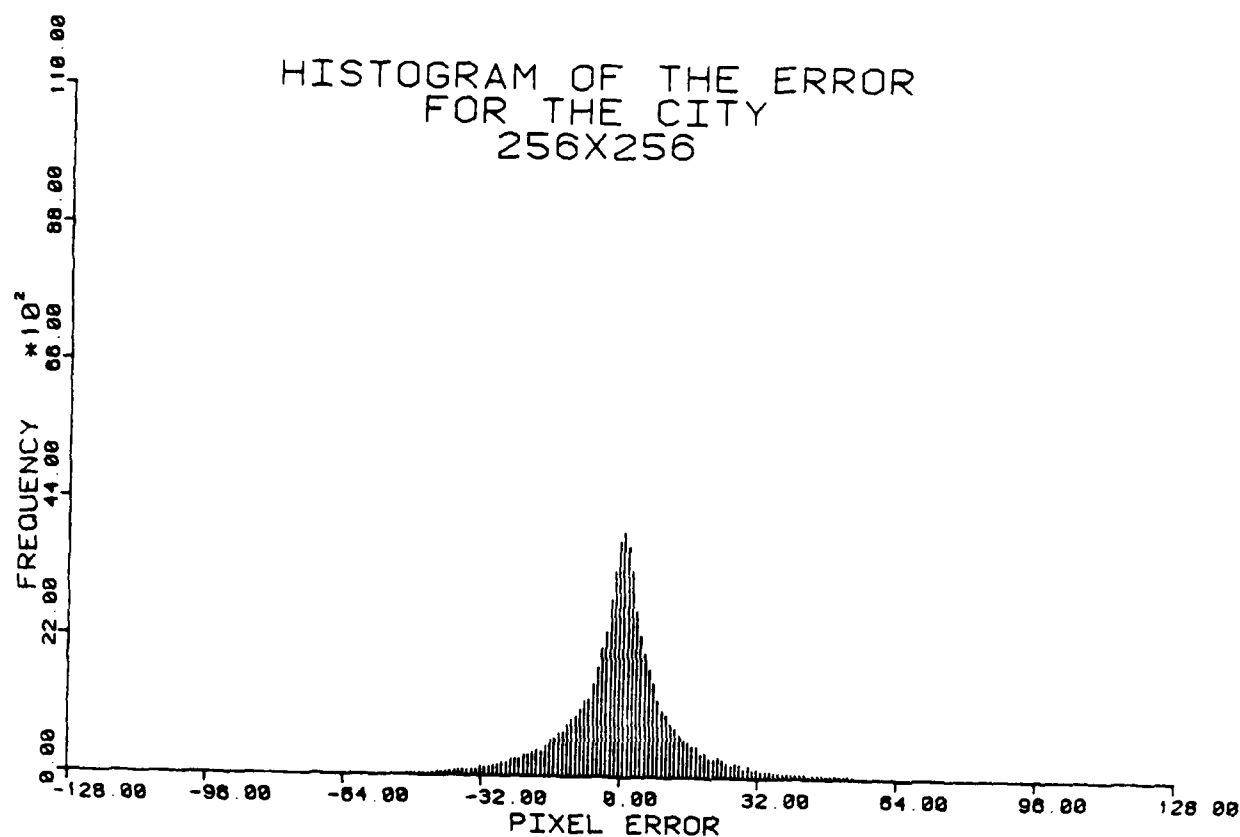
HISTOGRAM OF THE ERROR FOR THE CITY 256X256



ASPC / Kalman filter / Backward Adaptive Max
Data rate = 2.12 bits/pel SNR = 11.47 dB

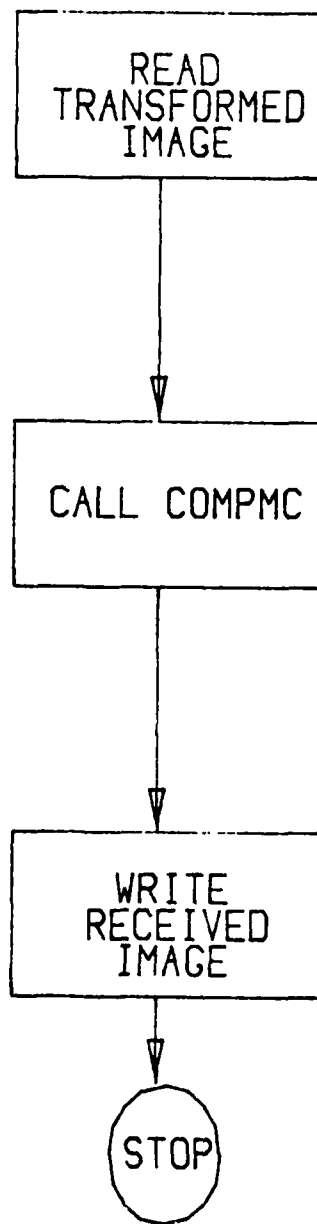


ASPC / Kalman filter / Backward Adaptive Max
Data rate = 1.62 bits/pel SNR = 9.514 dB
-ZEROING-

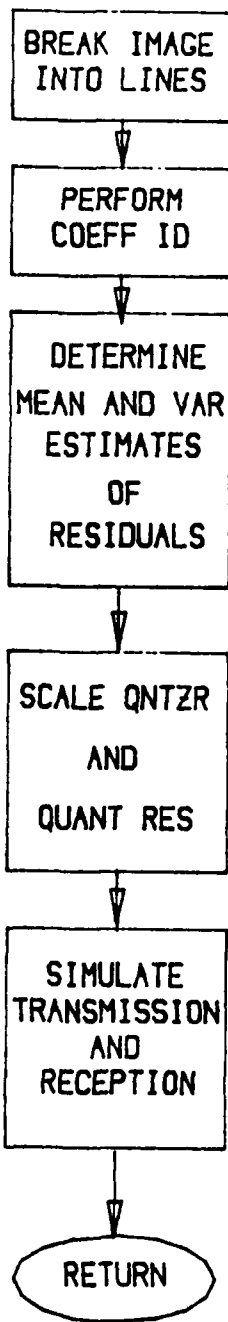


ASPC / Kalman filter / Backward Adaptive Max
Data rate = 1.62 bits/pel SNR = 9.514 dB
- ZEROING -

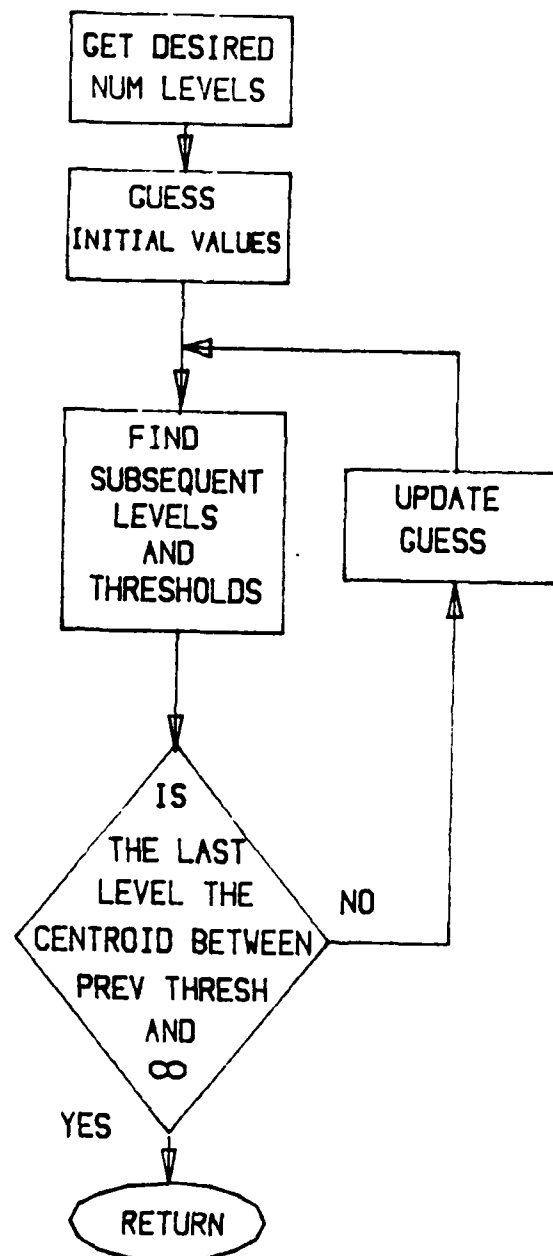
MAIN DRIVER



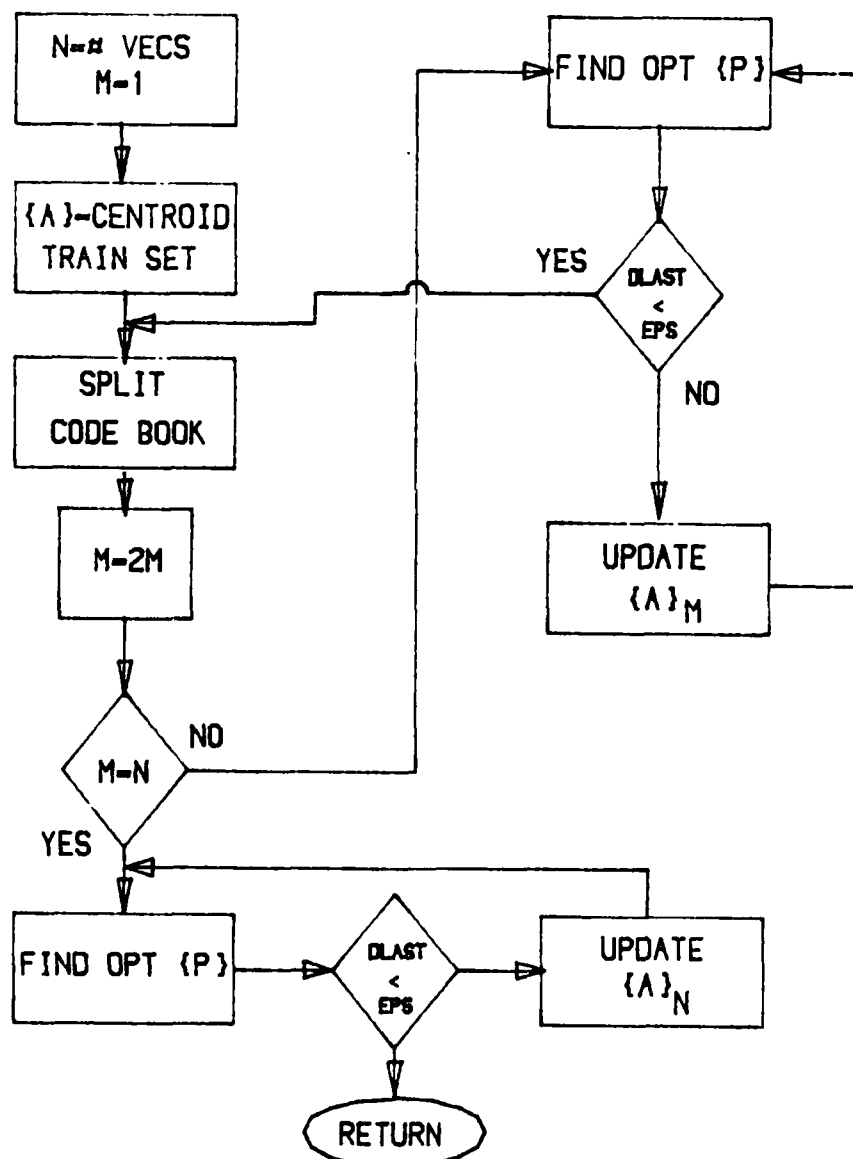
COMPMC



MAX



VECTOR



*** Main Driver Program for Transforms ***

```

EXTERNAL HADGEN, COSINE, HAARGN, KLGEN, FOURGN, DUMMY, SLAGEN
INTEGER HAD/4HHA /, HAAR/4HHAAR/, COS/4HCOS /, OTYPE, OBSZ,
1 SLAN/4HSLAN/, KL/4HKL /, FOUR/4HFOUR/, ID/4HID /
COMMON/CODPRM/ IR, IC, IBSZ, NTYPE, OBSZ, OTYPE, IBLKSZ
COMMON /CORE/ TEST(1024)
COMMON/FT/ KFFT
REAL XIN(256,256)
COMMON/B/ IDA, N
NN=16
ISIZE=16
KFFT=0
IBSZ=16
OBSZ=16
WRITE(6,200)
200 FORMAT('0',//, 'A A A A A A A A A A A A A A A A A A A A A A')
WRITE(6,201)
201 FORMAT('0',//)
DO 3000 KKK=1,1
READ(5,10) (ITYPE, M, N, IBLKSZ, IR, IC, NTYPE, LIPT, OTYPE, LOPT)
10 FORMAT(1X, A4, 1X, 2I5, 1X, I3, 1X, 2I2, 1X, A4, I3, 1X, A4, I3, 1X, I1)
IF(ITYPE.EQ.HAD) GO TO 100
IF(ITYPE.EQ.HAAR) GO TO 110
IF(ITYPE.EQ.COS) GO TO 120
IF(ITYPE.EQ.SLAN) GO TO 130
IF(ITYPE.EQ.KL) GO TO 140
IF(ITYPE.EQ.FOUR) GO TO 150
IF(ITYPE.EQ.ID) GO TO 160
20 WRITE(6,25) ITYPE
25 FORMAT(1H1, //, 133(1H?), //, 6X, A4, 1X, 'TRANSFORM NOT ALLOWED',
1 //, 133(1H?))
100 CALL XFORMB(HADGEN, M, N, LIPT, LOPT, XIN)
GO TO 170
110 CALL XFORMB(HAARGN, M, N, LIPT, LOPT, XIN)
GO TO 170
120 CALL XFORMB(COSINE, M, N, LIPT, LOPT, XIN)
GO TO 170
130 CALL XFORMB(SLAGEN, M, N, LIPT, LOPT, XIN)
GO TO 170
140 GO TO 20
150 KFFT=1
160 CALL XFORMB(DUMMY, M, N, LIPT, LOPT, XIN)
170 WRITE(6,180) ITYPE, IR, IC
180 FORMAT(1H1, //, 133(1H$), //, 45X, A4, 1X, 'TRANSFORM COMPLETED',
1 /, 45X, 'IR=', I2, /, 45X, 'IC=', I2, //, 133(1H$))
888 REWIND 3
IF(KKK.NE.1) GO TO 3000
DO 72 I=1,256
72 READ(3) (XIN(I, J), J=1,256)
REWIND 3
3000 CONTINUE

```

*** Main Driver Program for Transforms ***

STOP
END

*** Subroutine for Transformations ***

```

      SUBROUTINE XFORMB(RTN,M,N,LIPT,LOPT,DATA)
C*****
C  PURPOSE:  MAIN PROGRAM / TRANSFORM SUBROUTINES
C            INTERFACE.
C            RTN = TRANSFORMATION TO BE USED.
C            MXN = IMAGE DATA SIZE (M BY N)
C            LIPT = DATA LINE INPUT DISK UNIT NUMBER.
C            LOPT = DATA LINE OUTPUT DISK UNIT NUMBER.
C            DATA = IMAGE DATA
C*****
C  USER NOTE:  BE CERTAIN TO INCLUDE ALL NECESSARY COMMON
C              STATEMENTS IN MAIN PROGRAM.
C*****
      EXTERNAL RTN
      REAL A(1024),DATA(M,N)
      INTEGER OBSZ,OTYPE
      COMMON/FT/ KFFT
      COMMON/CODPRM/ IR,IC,IBSZ,NTYPE,OBSZ,OTYPE,IBLKSZ,IFLAG
      MAXRC = (1+KFFT)*IBLKSZ*N
      IF(MAXRC.GT.32800) GO TO 500
      IF(KFFT) 10,10,20
10    CALL BLKFRM(RTN,M,N,LIPT,LOPT,DATA)
      RETURN
20    CALL CBKFRM(RTN,M,N,LIPT,LOPT,IBLKSZ,DATA)
      RETURN
500   MXCORE =(MAXRC * 4)/1000
      WRITE(6,501) MXCORE
501   FORMAT(1H1,132(1H*),//,16X,115,'K REQUIRED, LINE BY LINE',
* 'NOT YET IMPLEMENTED',//,133(1H*))
      RETURN
      END

```

*** Subroutine to Break Image into Blocks ***

```

      SUBROUTINE BLKFRM(RTN,M,N,LIPT,LOPT,DATA)
C*****
C  PURPOSE:  IMAGE BLOCK FORMATION AND MANIPULATION.
C    RTN = TRANSFORMATION TO BE USED.
C    MXN = IMAGE SIZE (M BY N)
C    LIPT = LINE INPUT DISK UNIT NUMBER.
C    LOPT = LINE OUTPUT DISK UNIT NUMBER.
C    DATA = IMAGE DATA
C    IFLAG = 1 >>> KL TRANSFORM
C    IFLAG <> 1 >>> ALL OTHER TRANSFORMS
C*****
C  USER NOTE:  BE CERTAIN TO DUPLICATE ALL COMMON STATE-
C              MENTS.
C*****
      EXTERNAL RTN
      REAL DATA(IBLKSZ,N),A(1024)
      REAL*8 T(16,16)
      INTEGER OBSZ,OTYPE
      COMMON/CODPRM/ IR,IC,IBSZ,NTYPE,OBSZ,OTYPE,IBLKSZ,IFLAG
      COMMON/KLCOM/ T,IUNIT
      IF(IFLAG.EQ.1)REWIND IUNIT
      DO 50 LINE1=1,M,IBLKSZ
      DO 20 I=1,IBLKSZ
      LINE = LINE1 + I - 1
      CALL DSKIO(A,N,LINE,1,LIPT,NTYPE)
      DO 10 J=1,N
10    DATA(I,J) = A(J)
20    CONTINUE
      DO 30 IBLK=1,N,IBLKSZ
C*****
C  SOME SPECIAL I/O HANDLING OF EIGENVECTOR TRANS-
C  FORMATION IF KL TRANSFORMATION.  (IFLAG = 1)
C*****
      IF((IR.EQ.-1.OR.IC.EQ.-1).AND.(IFLAG.EQ.1))READ(IUNIT)T
      CALL RTN(DATA(1,IBLK),IBLKSZ,IBLKSZ,IR,IC,IBLKSZ,IBLKSZ)
30  IF((IR.EQ.1.OR.IC.EQ.1).AND.(IFLAG.EQ.1))WRITE(IUNIT)T
      DO 50 I=1,IBLKSZ
      LINE = LINE1 + I - 1
      DO 40 J=1,N
40    A(J) = DATA(I,J)
50    CALL DSKIO(A,N,LINE,0,LOPT,OTYPE)
      RETURN
      END

```

*** Subroutine to Handle Disk Input / Output ***

```

      SUBROUTINE DSKIO(A,N,LINE,RW,UNIT,TYPE)
C*****
C  PURPOSE:  READS AND WRITES INFORMATION TO AND FROM THE DISK
C            A LINE AT A TIME
C            A=DATA YOU WANT TOR READ OR WRITE
C            N=DATA VECTOR SIZE
C            LINE=DATA LINE NUMBER
C            RW=READ/WRITE FLAG
C            UNIT=DISK UNIT NUMBER FOR I/O OPERATION
C            TYPE=DATA TYPE(E.G. INTEGER*2,REAL*4...)
C*****
      INTEGER TB(100)/100*0/
      INTEGER RW,UNIT,TYPE,C4(1024),TEST2/4HIN*2/,TEST4/4HIN*4/
      INTEGER*2 B2(1024)
      REAL A(N)
      IF(LINE.GT.TB(UNIT)) GO TO 10
      TB(UNIT) = 0
      REWIND UNIT
10    TB(UNIT) = TB(UNIT) + 1
      IF(RW.EQ.0) GO TO 500
      IF(TYPE.NE.TEST2) GO TO 20
      READ(UNIT) (B2(I),I=1,N)
      DO 15 I=1,N
15    A(I) = B2(I)
      RETURN
20    IF(TYPE.NE.TEST4) GO TO 30
      READ(UNIT) (C4(I),I=1,N)
      DO 25 I=1,N
25    A(I) = C4(I)
      RETURN
30    READ(UNIT) (A(I),I=1,N)
      RETURN
500   IF(TYPE.NE.TEST2) GO TO 520
      DO 515 I=1,N
515   B2(I) = (A(I)+.5)
      WRITE(UNIT) (B2(I),I=1,N)
      RETURN
520   IF(TYPE.NE.TEST4) GO TO 530
      DO 525 I=1,N
525   C4(I) = A(I)
      WRITE(UNIT) (C4(I),I=1,N)
      RETURN
530   WRITE(UNIT) (A(I),I=1,N)
      RETURN
      END

```

*** Subroutine for Hadamard Transform ***

```

      SUBROUTINE HADGEN(C,MC,NC,IR,IC,LR,LC)
C
C      COMPUTES 2 DIM. HADAMARD TRANSFORM
C      OF MATRIX.
C      C = INPUT/OUTPUT MATRIX
C      MC=#OF ROWS OF C
C      NC=#OF COLUMNS OF C
C      IR = 0 FOR NO ROW TRANSFORM
C      IC = 0 FOR NO COLUMN TRANSFORM
C
C      REAL C(LR,LC)
C      COMMON /CORE/A(1024)
C
C      COLUMN TRANSFORM
C
C      IF ( IC .EQ. 0 ) GO TO 120
C      CALL POWER( MC , LMC )
C      DO 110 N=1,NC
C
C      FOR EACH COLUMN TRANSFORM ELEMENTS 1 THROUGH MC
C
C      DO 111 IZAP=1,1024
111  A(IZAP)=0.
C      DO 112 M=1,MC
C          A(M)=C(M,N)
112  CONTINUE
C          CALL FHT(LMC,A)
C          DO 115 M=1,MC
C              C(M,N)=A(M)
115  CONTINUE
110  CONTINUE
C
C      ROW TRANSFORM
C
120  IF ( IR .EQ. 0 ) RETURN
C      CALL POWER( NC , LNC )
C      DO 125 M=1,MC
C
C      COPY ROW M INTO ARRAY
C
C      DO 130 N=1,NC
130  A(N)=C(M,N)
C      CALL FHT(LNC,A)
C      DO 140 N=1,NC
140  C(M,N)=A(N)
125  CONTINUE
C      RETURN
C      END

```

*** Fast Hadamard Transform ***

```

SUBROUTINE FHT (M,X)
DIMENSION X(1)
N=2**M
NH=N/2
NHP=3*NH
NM=N-1
DO 100 L=1,M
LTEST=(L/2)*2
IF(LTEST.EQ.L) GO TO 200
JJ=0
JJN=NH
JK2=N
JK1=NM
GO TO 300
200 JJ=N
JJN=NHP
JK2=0
JK1=-1
300 DO 100 J=1,NH
JJ=JJ+1
JJN=JJN+1
JK1=JK1+2
JK2=JK2+2
X(JK1) = X(JJ) + X(JJN)
X(JK2) = X(JJ) - X(JJN)
100 CONTINUE
MTEST=(M/2)*2
IF(MTEST.EQ.M) GO TO 400
DO 500 I=1,N
500 X(I)=X(N+I)
400 RETURN
END

```

*** Fast Cosine Transform ***

```

      SUBROUTINE FCT(INPUT, ISIZE, ITYPE)
C
C   *
C   THIS SUBROUTINE COMPUTES THE COSINE TRANSFORM OF DATA STORED
C   IN AN INPUT ARRAY AND RETURNS THE RESULTS IN THE SAME ARRAY.
C   THERE IS A CHECK MADE SO THE TABLE IS ONLY CALCULATED THE
C   FIRST TIME A CALL IS MADE, OR IF ISIZE CHANGES.
C   MAXIMUM SIZE ARRAY = 256 ELEMENTS
C
C   INPUT = INPUT/OUTPUT ARRAY
C   ISIZE = NUMBER OF ELEMENTS IN INPUT/OUTPUT ARRAY
C   ITYPE:  1= FORWARD TRANSFORM
C           -1= INVERSE TRANSFORM
C
      REAL S(128)
      REAL COSINE(255), SINE(255), INPUT(512), OUTPUT(1024)
      INTEGER INV(128), MM(3), OISIZE, OTYPE
      DATA OTYPE/0/
      DATA MM/9,0,0/
      DATA ICHECK/0/
      IF(ICHECK.NE.0.AND.ISIZE.EQ.OISIZE) GOTO 100
      ICHECK=1
      OISIZE=ISIZE
      ISIZEF=ISIZE*4
      SQRT2=SQRT(2.0)
      CALL POWER(ISIZE, MM(1))
      MM(1)=MM(1)+1
C
C   COSINE TRANSFORM TABLE GENERATION
C
      Y=3.1415927/2.0/FLOAT(ISIZE)
      CC=COS(Y)
      SC=SIN(Y)
      COSINE(1)=CC
      SINE(1)=SC
      JJ=ISIZE-2
      DO 950 I=1, JJ
      COSINE(I+1)=COSINE(I)*CC - SINE(I)*SC
950   SINE(I+1)=SINE(I)*CC + COSINE(I)*SC
C   END COSINE TRANSFORM TABLE GENERATION
C
C   BRANCH TO DO FORWARD OR INVERSE TRANSFORM
100   IF(ITYPE) 888, 888, 999
C
C   FORWARD TRANSFORM
C
999   OUTPUT(1)=INPUT(1)
      OUTPUT(2)=0.0
      DO 900 I=2, ISIZE
      OUTPUT(I*2-1)=INPUT(I)
      OUTPUT(ISIZEF+3-I*2)=0.0

```

*** Subroutine for Cosine Transform ***

```

      SUBROUTINE COSINE(C,MC,NC,IR,IC,LR,LC)
C
C      C=INPUT / OUTPUT MATRIX
C      MC=NUMBER OF ROWS OF C TO BE TRANSFORMED
C      NC=NUMBER OF COLUMNS OF C TO BE TRANSFORMED
C      IR=1, FORWARD ROW TRANSFORM
C      IR=0, NO ROW TRANSFORM
C      IR=-1, INVERSE ROW TRANSFORM
C      IC=1, FORWARD COLUMN TRANSFORM
C      IC=0, NO COLUMN TRANSFORM
C      IC=-1, INVERSE COLUMN TRANSFORM
C
      COMMON /CORE/ A(1024)
      REAL C(LR,LC)
      IF (LR.EQ. 0) GO TO 140
C
C      REPEAT FOR EACH COLUMN
C
      DO 110 N=1,NC
C
C      C(1,N) POINTS TO ROW 1 COLUMN N
C
      DO 112 M=1,MC
      A(M)=C(M,N)
112  CONTINUE
      CALL FCT(A,MC,IC)
      DO 115 M=1,MC
      C(M,N)=A(M)
115  CONTINUE
110  CONTINUE
140  IF (IR.EQ. 0) RETURN
C
C      FOR EACH ROW COPY TO DUMMY ARRAY
C
      DO 150 M=1,MC
C
C      COPY ROW M INTO A
C
      DO 130 N=1,NC
130  A(N)=C(M,N)
      CALL FCT(A,NC,IR)
      DO 145 N=1,NC
145  C(M,N)=A(N)
150  CONTINUE
      RETURN
      END

```

*** Fast Cosine Transform ***

```

      OUTPUT(ISIZEF+4-I*2)=0.0
900  OUTPUT(I*2)=0.0
      OUTPUT(2*ISIZE+1)=0.
      OUTPUT(2*ISIZE+2)=0.
      CALL HARM(OUTPUT,MM,INV,S,1,IFERR)
      IF(IFERR.NE.0) GOTO 998
      INPUT(1)=OUTPUT(1)*SQRT2
      DO 810 I=2, ISIZE
810  INPUT(I)=2.0*((OUTPUT(I*2-1)*COSINE(I-1))-
      *OUTPUT(I*2)*SINE(I-1))
      GOTO 1001

C
C INVERSE TRANSFORM
C
888  SUM=INPUT(1)/SQRT2
      CT=SUM*(1.0-1.0/SQRT2)/FLOAT(ISIZE)
      OUTPUT(1)=INPUT(1)
      OUTPUT(2)=0.0
      DO 910 I=2, ISIZE
      OUTPUT(I*2-1)=INPUT(I)*COSINE(I-1)
      OUTPUT(ISIZEF+3-I*2)=INPUT(I)*COSINE(I-1)
      OUTPUT(ISIZEF+4-I*2)= INPUT(I)*SINE(I-1)
910  OUTPUT(I*2)=-(INPUT(I)*SINE(I-1))
      OUTPUT(2*ISIZE+1)=0.
      OUTPUT(2*ISIZE+2)=0.
      CALL HARM(OUTPUT,MM,INV,S,-1,IFERR)

C
      IBM FFT ROUTINE (3D.)
      IF(IFERR.EQ.0) GOTO 960
998  IF(OTYPE.NE.0)WRITE(6,612) IFERR
612  FORMAT(' ','HARM ERROR',' * ERROR= ',I2)
      STOP 777
960  CONTINUE
      DO 811 I=1, ISIZE
811  INPUT(I)=OUTPUT(I*2-1)+CT
1001 CONTINUE
      RETURN
      END

```

*** Subroutine to Determine Power ***

```

      SUBROUTINE POWER(JJ,IPOWER)
C**  THIS SUBROUTINE INPUTS A VALUE 'JJ' AND RETURNS
C**  ITS POWER OF 2 - 'IPOWER'.
C**  WHERE:
C**              IPOWER
C**      JJ      =  2
C**
      INTEGER OTYPE
      DATA OTYPE/0/
      DATA IOUT/6/
C  IOUT IS THE PRINTER UNIT NUMBER.
      NC=JJ
      IPOWER=0
10     I=NC/2
      NC=I
      IPOWER=IPOWER+1
      IF(I.GT.1) GOTO 10
      IF(I.EQ.1) GOTO 20
      IF(OTYPE.NE.0)WRITE(IOUT,1)
1     FORMAT(' ','***** POWER ERROR ***** INPUT IS NOT A POWER OF 2')
      STOP 777
20     RETURN
      END

```

*** Main Driver Program for all ASPC Simulations ***

```

      INTEGER IN2/4HIN*2/,IN4/4HIN*4/,RL/4HREAL/
      REAL*4 XIN(256,256),COEFF(7,256)
      REAL*4 QTH(50),QL(50)
      REAL A(256)

```

```

C-----
C
C
C
C  AHPC/ASPC MAIN DRIVER PROGRAM
C
C  EXPLANATIONS OF VARIABLES:
C    XIN      - MAIN I/O IMAGE MATRIX
C    COEFF    - COEFFICIENT STORAGE MATRIX
C    JBLKSZ   - TRANSFORMATION BLOCKSIZE
C    ISZ      - IMAGE ROW SIZE
C    JSZ      - IMAGE COLUMN SIZE
C    NUMSMP   - NUMBER SAMPLES/LINE (SAME AS JSZ)
C    NUMFR    - NUMBER SAMPLE LINES/FRAME (SAME AS ISZ)
C    QTH      - QUANTIZER THRESHOLDS
C    QL       - QUANTIZER LEVELS
C
C  SUBROUTINES:
C    DSKIO    - DISK INPUT / OUTPUT
C    MAXQTZ   - MAX QUANTIZER DESIGN
C    COMPMC   - ADPCM DATA COMPRESSION
C-----

```

```

      NUMFR=256
      NUMSMP=256
      NCOEFF=3
      ISZ=256
      JSZ=256
      JBLKSZ=16
      COMMON /A/ JBLKSZ
      DO 10 I=1,ISZ
      CALL DSKIO(A,JSZ,I,1,1,RL)
      DO 10 J=1,JSZ
10  XIN(I,J)=A(J)
      DO 11 I=1,12
11  WRITE(6,1001)(XIN(I,J),J=1,16)
1001 FORMAT(' ',16(F6.0,1X))
      NBIT=1
      CALL MAXQTZ(QTH,QL,NLEVEL,NBIT)
      CALL COMPMC(XIN,NUMFR,NUMSMP,NCOEFF,COEFF,QTH,QL,NLEVEL)
      WRITE(6,20)
20  FORMAT(' ***** MADE IT BACK THROUGH SUBS *****')
      DO 30 I=1,12
30  WRITE(6,1001)(XIN(I,J),J=1,16)
      DO 40 I=1,ISZ
      DO 35 J=1,JSZ

```

*** Main Driver Program for all ASFC Simulations ***

```
35 A(J)=XIN(I,J)
   CALL DSKIO(A,JSZ,I,0,2,RL)
40 CONTINUE
   WRITE(6,2)
  2 FORMAT('0*** JOB FINISHED ***')
   STOP
   END
```

*** Subroutine for ADPCM using ARMA Model ***

```
SUBROUTINE COMPMC(XIN,NF,NSF,N,CM,QTH,QL,NLEVEL)
REAL A(10),XIN(NF,NSF),R(7),XV(256),XTR(256,256)
REAL CM(7,256),QL(1),QTH(1),E2V(256),VAREST(256),XMN(256)
REAL ACV(256),AC(10),PACV(10),WORK(10)
REAL P(10),WW(150),RESMN(256)
REAL B(10,10),PAR(10)
REAL QML(1024),QMT(1024)
INTEGER NARPS(256)
```

EXPLANATION OF CALL VARIABLES:

```
XIN      - THE INPUT DATA MATRIX AND RESIDUAL OUTPUT MATRIX
NF       - NUMBER OF FRAMES (NUMBER OF LINES IN THE INPUT MATRIX)
NSF      - NUMBER OF SAMPLES PER FRAME (SAMPLES PER LINE)
N        - NUMBER OF PREDICTOR COEFFICIENTS TO BE USED
CM       - MATRIX THAT CONTAINS THE PREDICTOR COEFFICIENT VECTORS
```

SET UP THE CONSTANTS AND INITIAL VALUES

```
CALL MAXQTZ(QMT,QML,MLEVEL,6)
XNSF = NSF
COMMON /A/ JBLKSZ
XAVBLK=JBLKSZ*128.
IP=N
IQ=0
NHOLD=N
```

IN THE DO 160 LOOP, I IS THE LINE NUMBER (1 - NF)

```
VSME=0.
VSMSE=0.
RSME=0.
RSMSE=0.
NLEV=2*NLEVEL
DO 160 I=1,NF
N=NHOLD
SUM=0.
SUMSX=0.
DO 501 J=1,NSF
501 XV(J)=XIN(I,J)
DO 510 J=1,NSF
SUM=SUM+XV(J)
SUMSX=SUMSX+XV(J)*XV(J)
510 CONTINUE
XMN(I)=SUM/NSF
IF(MOD(I-1,JBLKSZ) .NE. 0)GO TO 373
XMN(I)=2750.
GO TO 374
373 XMN(I)=0.
374 CONTINUE
```

*** Subroutine for ADPCM using ARMA Model ***

```

321 CALL FTAUTO(XV,NSF,N,N,7,XMN(I),ACV(1),ACV(2),AC,PACV,WORK)
    IP=N
    IQ=0
    CALL FTARPS(ACV,XBAR,IP,IQ,A,PMAC,WW,IER)
    IF(N.EQ. 2)GO TO 432
    IF(IER.NE. 129)GO TO 432
    N=N-1
    WRITE(6,636)N
636 FORMAT('      ---> N MADE TO: ',I3)
    GO TO 321
432 CALL PARSBL(A,N,PAR,B)
    DO 335 KK=1,N
    CM(KK,I)=UNIFRM(-1.75,1.75,MLEVEL,A(KK))
335 CONTINUE
    DO 700 II=1,N
700 R(II)=XMN(I)
    DO 710 II=1,NSF
    PRED=0.
    DO 705 J=1,N
705 PRED=CM(J,I)*R(J)+PRED
    PRED=PRED+PMAC
    E2V(II)=XV(II)-PRED
    DO 706 K=2,N
706 R(N+2-K)=R(N+1-K)
    R(1)=XV(II)
710 CONTINUE
    SUME=0.
    SUMSQE=0.
    DO 155 M=1,NSF
    SUME=SUME+E2V(M)
155 SUMSQE=SUMSQE+(E2V(M)*E2V(M))
    VAREST(I)=(SUMSQE-((SUME*SUME)/NSF))/(NSF-1.)
    VSME=VSME+VAREST(I)
    VSMSE=VSMSE+VAREST(I)*VAREST(I)
    RESMN(I)=SUME/NSF
    WRITE(6,919)I,RESMN(I),VAREST(I),(A(K1),K1=1,N),(CM(K2,I),K2=1,N)
919 FORMAT(' ',I4,5(1X,E10.3),/,3(1X,E10.3))
    VAREST(I)=UNIFRM(0.,40000.,512,VAREST(I))
    IF(MOD(I-1,JBLKSZ).EQ. 0)RESMN(I)=RESMN(I)/JBLKSZ
    RSME=RSMN+RESMN(I)
    RSMSE=RSMSE+RESMN(I)*RESMN(I)
    RESMN(I)=QNTZ(RESMN(I),MLEVEL,QML,QMTH,150.)
    IF(MOD(I-1,JBLKSZ).EQ. 0)RESMN(I)=RESMN(I)*JBLKSZ
    DO 200 J=1,N
200 R(J) = XMN(I)
    DO 250 J=1,NSF
    S=XV(J)
    RT2=0.0

```

C

C NOW GET XMITTER RESIDUAL

*** Subroutine for ADPCM using ARMA Model ***

C

```
DO 230 K=1,N
230 RT2=RT2 + CM(K,I)*R(K)
E2 = S - RT2 - RESMN(I)
EQ2=QNTZ(E2,NLEVEL,QL,QTH,VAREST(I))
XTR(I,J)=EQ2
```

C

C

C

C

C

SHIFT R-REGISTER

```
DO 240 K=2,N
240 R(N+2-K) = R(N+1-K)
R(1) = RT2+EQ2+RESMN(I)
250 CONTINUE
NARPS(I)=N
160 CONTINUE
VMN=VSME/256.
VVAR=(VSMSE-((VSME*VSME)/256.))/255.
RMN=RSME/256.
RVAR=(RSMSE-((RSME*RSME)/256.))/255.
WRITE(6,920)VMN,VVAR,RMN,RVAR
920 FORMAT(' VMN=',E15.5,' VV ',E15.5,' RMN=',E15.5,' RVAR=',E15.5)
```

C

C

C

C

C

RECEIVER SIMULATION SECTION

C

C

C

C

C

SET R - REGISTER TO MEAN VALUE
FOR THE CURRENT LINE.

```
DO 210 I = 1,NF
N=NARPS(I)
DO 170 J = 1,N
170 R(J)=XMN(I)
DO 400 J = 1,NSF
RV2 = 0.0
DO 180 K = 1,N
180 RV2 = RV2 + CM(K,I)*R(K)
RN = XTR(I,J) + RV2 + RESMN(I)
XIN(I,J)=RN
```

C

C

C

C

C

SHIFT R - REGISTER.

```
DO 190 K = 2,N
190 R(N+2-K) = R(N+1-K)
```

*** Subroutine for ADPCM using ARMA Model ***

R(1) = RN
400 CONTINUE
210 CONTINUE
RETURN
END

*** Subroutine for ADPCM using Kalman Filter ***

```
SUBROUTINE COMPMC(XIN,NF,NSF,N,CM,QTH,QL,NLEVEL)
REAL A(10),V1(10),XIN(256,256),R(10),XV(256),G(10),V(10,10)
REAL XTR(256,256)
REAL CM(7,256),QL(1),QTH(1),E2V(256),VAREST(256),XMEAN(256)
REAL RESMN(256)
REAL QML(1024),QMT(1024)
REAL B(10,10),PAR(10)
```

EXPLANATION OF CALL VARIABLES:

```
XIN - THE INPUT DATA MATRIX AND RESIDUAL OUTPUT MATRIX
NF - NUMBER OF FRAMES (NUMBER OF LINES IN THE INPUT MATRIX)
NSF - NUMBER OF SAMPLES PER FRAME (SAMPLES PER LINE)
N - NUMBER OF PREDICTOR COEFFICIENTS TO BE USED
CM - MATRIX THAT CONTAINS THE PREDICTOR COEFFICIENT VECTORS
```

DEFINITION OF VARIABLE TERMS:

```
V - THE ERROR COVARIANCE MATRIX
A - THE PREDICTOR COEFFICIENT VECTOR
VARI - INITIAL VALUE FOR THE ERROR COVARIANCE MATRIX
VV - VARIANCE OFFSET
XV - THE INPUT LINE TEMPORARY VECTOR
G - THE GAIN VECTOR
R - THE PAST VALUE VECTOR
E - THE ERROR OR RESIDUAL TERM
```

SET UP THE CONSTANTS AND INITIAL VALUES

```
DATA V/100*0.0/,A/1.0,-.5,-.2,.3,.4,-.5/
CALL MAXQTZ(QMT,QML,MLEVEL,6)
VLO=9999999.
VHIGH=-1.
XNSF = NSF
COMMON /A/ JBLKSZ
XAVBLK=JBLKSZ*128.
NSFM1=NSF-1
VV = 1.0
```

IN THE DO 160 LOOP, I IS THE LINE NUMBER (1 - NF)

```
DO 160 I=1,NF
VARI=100.
DO 10 J=1,N
DO 10 K=1,N
V(J,K) = 0.0
IF(J.EQ.K) V(J,K) = VARI
10 CONTINUE
```

SET UP THE INPUT VECTOR AND THE PAST VALUE VECTOR

```
SUMX=0.
```

*** Subroutine for ADPCM using Kalman Filter ***

```

      DO 20 J=1,NSF
      SUMX=SUMX+XIN(I,J)
20   XV(J) = XIN(I,J)
      XMEAN(I)=SUMX/XNSF
      WRITE(6,2000)I,XMEAN(I)
2000 FORMAT(' LINE: ',I5,2X,E15.5)
      IF(MOD(I-1,JBLKSZ) .NE. 0)GO TO 373
      XMEAN(I)=2750.
      GO TO 374
373  XMEAN(I)=0.
374  DO 30 J=1,N
30   R(J) = XMEAN(I)

```

C
C
C
C

IDENTIFICATION LOOP (IDENTIFY THE PREDICTOR COEFFICIENTS)

```

      DO 100 J=1,NSF
      S = XV(J)
      S2 = 0.0
      RT = 0.0
      DO 40 K=1,N
      V1(K) = 0.
      DO 40 L=1,N
40   V1(K) = V1(K) + V(K,L)*R(L)
      DO 50 K=1,N
      S2 = S2 + R(K)*V1(K)
      DO 60 K=1,N
      G(K) = V1(K)/(VARI + S2)
60   RT = RT + A(K)*R(K)
      DO 70 K=1,N
      DO 70 L=1,K
      V(K,L) = V(K,L) - G(K)*V1(L)
70   V(L,K) = V(K,L)
      E = S - RT
      DO 80 K=1,N
80   A(K) = A(K) + G(K)*E

```

C
C
C

SHIFT THE PAST VALUE VECTOR

```

      DO 90 L=2,N
90   R(N+2-L) = R(N+1-L)
      R(1) = RT+E
      IF(I .EQ. 225)WRITE(6,1001)(G(K1),K1=1,N)
1001 FORMAT(' G=',3(1X,E15.5))
100  CONTINUE
      CALL PARABL(A,N,PAR,B)
      DO 335 KK=1,N
      CM(KK,I)=UNIFRM(-1.75,1.75,256,A(KK))
335  CONTINUE

```

C

*** Subroutine for ADPCM using Kalman Filter ***

C RESET R-REGISTER TO INITIAL SIGNAL VALUE
C

DO 120 J=1,N
120 R(J) = XMEAN(I)
DO 150 J=1,NSF
S = XV(J)
RT2 = 0.0

C
C DETERMINE RESIDUAL SIGNAL
C

C RT2 IS AT THE TRANSMITTER STAGE WHERE:
C RT2 = RT2 + AO(K)*RV(K)
C

DO 130 K=1,N
130 RT2 = RT2 + CM(K,I)*R(K)
E2V(J)=S-RT2

C
C SHIFT R-REGISTER
C

DO 140 K=2,N
140 R(N+2-K) = R(N+1-K)
R(1) = RT2+E2V(J)
150 CONTINUE
SUME=0.
SUMSQE=0.
DO 155 M=1,NSF
SUME=SUME+E2V(M)
155 SUMSQE=SUMSQE+(E2V(M)*E2V(M))
VAREST(I)=(SUMSQE-(SUME*SUME/NSF))/NSFM1
IF(VAREST(I) .LT. VLO)VLO=VAREST(I)
IF(VAREST(I) .GT. VHIGH)VHIGH=VAREST(I)
VSME=VSME+VAREST(I)
VSMSE=VSMSE+VAREST(I)*VAREST(I)
RESMN(I)=SUME/XNSF
IF(MOD(I-1,JBLKSZ) .EQ. 0)RESMN(I)=RESMN(I)/JBLKSZ
WRITE(6,919)I,RESMN(I),VAREST(I),(A(K1),K1=1,N),(CM(K2,I),K2=1,N)
919 FORMAT(' ',I4,5(1X,E10.3),/,3(1X,E10.3))
VAREST(I)=UNIFORM(0.,40000.,512,VAREST(I))
RSME=RSME+RESMN(I)
RSMSE=RSMSE+RESMN(I)*RESMN(I)
RESMN(I)=QNTZ(RESMN(I),MLEVEL,QML,QMTH,150.)
IF(MOD(I-1,JBLKSZ) .EQ. 0)RESMN(I)=RESMN(I)*JBLKSZ
DO 200 J=1,N
200 R(J) = XMEAN(I)
DO 250 J=1,NSF
S=XV(J)
RT2=0.0

C
C NOW GET XMITTER RESIDUAL
C

*** Subroutine for ADPCM using Kalman Filter ***

```

      DO 230 K=1,N
230  RT2=RT2 + CM(K,I)*R(K)
      E2 = S - RT2 - RESMN(I)
      EQ2=QNTZ(E2,NLEVEL,QL,QTH,VAREST(I))
      XTR(I,J)=EQ2

```

C
C
C
C
C

SHIFT R-REGISTER

```

      DO 240 K=2,N
240  R(N+2-K) = R(N+1-K)
      R(1) = RT2+EQ2+RESMN(I)
250  CONTINUE
160  CONTINUE
      VMN=VSME/256.
      VVAR=(VSMSE-((VSME*VSME)/256.))/255.
      RMN=RSME/256.
      RVAR=(RSMSE-((RSME*RSME)/256.))/255.
      WRITE(6,920) VMN,VVAR,RMN,RVAR
920  FORMAT(' VMN=',E12.5,' VVAR=',E12.5,' RMN=',E12.5,' RVAR=',E12.5)
      WRITE(6,921) VLO,VHIGH
921  FORMAT(' VLO=',E15.5,' VHIGH=',E15.5)

```

C
C
C
C
C
C
C
C
C
C
C
C

RECEIVER SIMULATION SECTION

SET R - REGISTER TO MEAN VALUE
FOR THE CURRENT LINE.

```

      DO 210 I = 1,NF
      DO 170 J = 1,N
170  R(J)=XMEAN(I)
      DO 400 J = 1,NSF
      S = XIN(I,J)
      RV2 = 0.0
      DO 180 K = 1,N
180  RV2 = RV2 + CM(K,I)*R(K)
      RN = XTR(I,J) + RV2 + RESMN(I)
      XIN(I,J) = RN

```

C
C
C
C
C

SHIFT R - REGISTER.

```

      DO 190 K = 2,N
190  R(N+2-K) = R(N+1-K)

```

*** Subroutine for ADPCM using Kalman Filter ***

R(1) = RN
400 CONTINUE
210 CONTINUE
RETURN
END

*** Subroutine for ADPCM using Stoch. Approx. ***

```
SUBROUTINE COMPMC(XIN,NF,NSF,N,CM,QTH,QL,NLEVEL)
REAL A(10),XIN(NF,NSF),R(7),XV(256),XTR(256,256)
REAL CM(7,NF),QL(1),QTH(1),E2V(256),VAREST(256),XMN(256)
REAL RESMN(256)
REAL B(10,10),PAR(10)
REAL QML(1024),QMT(1024)
```

EXPLANATION OF CALL VARIABLES:

```
XIN  - THE INPUT DATA MATRIX AND RESIDUAL OUTPUT MATRIX
NF   - NUMBER OF FRAMES (NUMBER OF LINES IN THE INPUT MATRIX)
NSF  - NUMBER OF SAMPLES PER FRAME (SAMPLES PER LINE)
N    - NUMBER OF PREDICTOR COEFFICIENTS TO BE USED
CM   - MATRIX THAT CONTAINS THE PREDICTOR COEFFICIENT VECTORS
```

IN THE DO 160 LOOP, I IS THE LINE NUMBER (1 - NF)

```
CALL MAXQTZ(QMT,QML,MLEVEL,6)
COMMON /A/ JBLKSZ
XAVBLK=JBLKSZ*256.
DO 160 I=1,NF
DO 501 J=1,NSF
501 XV(J)=XIN(I,J)
CALL STAPRX(XV,NSF,N,A,XMN(I))
IF(MOD(I-1,JBLKSZ) .NE. 0) GO TO 373
XMN(I)=2750.
GO TO 374
373 XMN(I)=0.
374 CALL PARSBL(A,N,PAR,B)
DO 335 KK=1,N
CM(KK,I)=UNIFORM(-1.75,1.75,256,A(KK))
335 CONTINUE
DO 700 II=1,N
700 R(II)=XMN(I)
DO 710 II=1,NSF
PRED=0.
DO 705 J=1,N
705 PRED=CM(J,I)*R(J)+PRED
E2V(II)=XV(II)-PRED
DO 706 K=2,N
706 R(N+2-K)=R(N+1-K)
R(1)=XV(II)
710 CONTINUE
SUME=0.
SUMSQE=0.
DO 155 M=1,NSF
SUME=SUME+E2V(M)
155 SUMSQE=SUMSQE+(E2V(M)*E2V(M))
VAREST(I)=(SUMSQE-((SUME*SUME)/NSF))/(NSF-1.)
```

*** Subroutine for ADPCM using Stoch. Approx. ***

```

RESMN(I)=SUME/NSF
IF(MOD(I-1,JBLKSZ) .EQ. 0) RESMN(I)=RESMN(I)/JBLKSZ
WRITE(6,919) I, RESMN(I), VAREST(I), (A(K1), K1=1, N), (CM(K2, I), K2=1, N)
919 FORMAT(' ', I4, 5(1X, E10.3), /, 3(1X, E10.3))
VAREST(I)=UNIFRM(0., 100000., 512, VAREST(I))
RESMN(I)=QNTZ(RESMN(I), MLEVEL, QML, QMTH, 150.)
IF(MOD(I-1,JBLKSZ) .EQ. 0) RESMN(I)=RESMN(I)*JBLKSZ
DO 200 J=1, N
200 R(J) = XMN(I)
DO 250 J=1, NSF
S=XV(J)
RT2=0.0
C
C NOW GET XMITTER RESIDUAL
C
DO 230 K=1, N
230 RT2=RT2 + CM(K, I)*R(K)
E2 = S - RT2 - RESMN(I)
EQ2=QNTZ(E2, NLEVEL, QL, QTH, VAREST(I))
XTR(I, J)=EQ2
C
C SHIFT R-REGISTER
C
DO 240 K=2, N
240 R(N+2-K) = R(N+1-K)
R(1) = RT2+EQ2+RESMN(I)
250 CONTINUE
160 CONTINUE
C-----
C
C RECEIVER SIMULATION SECTION
C-----
C
C SET R - REGISTER TO MEAN VALUE
C FOR THE CURRENT LINE.
C
C
DO 210 I = 1, NF
DO 170 J = 1, N
170 R(J)=XMN(I)
DO 400 J = 1, NSF
RV2 = 0.0
DO 180 K = 1, N
180 RV2 = RV2 + CM(K, I)*R(K)
RN = XTR(I, J) + RV2 + RESMN(I)
XIN(I, J)=RN
C-----

```

*** Subroutine for ADPCM using Stoch. Approx. ***

C
C
C
C

SHIFT R - REGISTER.

DO 190 K = 2,N
190 R(N+2-K) = R(N+1-K)
R(1) = RN
400 CONTINUE
210 CONTINUE
RETURN
END

*** Subroutine to Provide PARCOR Stabilization ***

```

      SUBROUTINE PARSBL(A,N,PAR,B)
C-----
C
C   COEFFICIENT PARCOR STABILIZATION
C   SUBROUTINE
C
C
      REAL A(1),B(10,10),R(13),PAR(1)
10  DO 100 I=1,N
100  B(I,N)=-A(I)
      IF(ABS(A(N)) .GT. 0.97)GO TO 500
      PAR(N)=-A(N)
      NM1=N-1
      DO 300 I=1,NM1
      K=N+1-I
      KM=K-1
      G=1.0/(1.0-B(K,K)*B(K,K))
      DO 200 KK=1,KM
200  B(KK,KM)=(B(KK,K)-B(K,K)*B(K-KK,K))*G
      IF(ABS(B(KM,KM)) .GT. 0.97)GO TO 500
      PAR(KM)=B(KM,KM)
300  CONTINUE
      RETURN
500  GG=1.0
      DO 510 I=1,N
      GG=0.97*GG
510  A(I)=GG*A(I)
      GO TO 10
      END

```

*** Subroutine for ADPCM / Kalman filter / Zeroing ***

```
SUBROUTINE COMPMC(XIN,NF,NSF,N,CM,QTH,QL,NLEVEL)
REAL A(10),V1(10),XIN(256,256),R(10),XV(256),G(10),V(10,10)
REAL XTR(256,256)
REAL CM(7,256),QL(1),QTH(1),E2V(256),VAREST(256),XMEAN(256)
REAL RESMN(256)
REAL QML(1024),QMT(1024)
REAL B(10,10),PAR(10)
```

EXPLANATION OF CALL VARIABLES:

```
XIN  - THE INPUT DATA MATRIX AND RESIDUAL OUTPUT MATRIX
NF   - NUMBER OF FRAMES (NUMBER OF LINES IN THE INPUT MATRIX)
NSF  - NUMBER OF SAMPLES PER FRAME (SAMPLES PER LINE)
N    - NUMBER OF PREDICTOR COEFFICIENTS TO BE USED
CM   - MATRIX THAT CONTAINS THE PREDICTOR COEFFICIENT VECTORS
```

DEFINITION OF VARIABLE TERMS:

```
V    - THE ERROR COVARIANCE MATRIX
A    - THE PREDICTOR COEFFICIENT VECTOR
VARI - INITIAL VALUE FOR THE ERROR COVARIANCE MATRIX
VV   - VARIANCE OFFSET
XV   - THE INPUT LINE TEMPORARY VECTOR
G    - THE GAIN VECTOR
R    - THE PAST VALUE VECTOR
E    - THE ERROR OR RESIDUAL TERM
```

SET UP THE CONSTANTS AND INITIAL VALUES

```
DATA V/100*0.0/,A/1.0,-.5,-.2,.3,.4,-.5/
LIMCNT=2
IBLAT=1
CALL MAXQTZ(QMT,QML,MLEVEL,6)
VLO=9999999.
VHIGH=-1.
XNSF = NSF
COMMON /A/ JBLKSZ
XAVBLK=JBLKSZ*128.
NSFM1=NSF-1
VV = 1.0
VARI = 100.0
```

IN THE DO 160 LOOP, I IS THE LINE NUMBER (1 - NF)

```
DO 160 I=1,NF
DO 10 J=1,N
DO 10 K=1,N
V(J,K) = 0.0
IF(J.EQ.K) V(J,K) = VARI
```

*** Subroutine for ADPCM / Kalman filter / Zeroing ***

```

10  CONTINUE
C
C      SET UP THE INPUT VECTOR AND THE PAST VALUE VECTOR
C
      SUMX=0.
      DO 20 J=1,NSF
      SUMX=SUMX+XIN(I,J)
20  XV(J) = XIN(I,J)
      IF(MOD(I-1,JBLKSZ) .NE. 0)GO TO 373
      XMEAN(I)=2750.
      GO TO 374
373 XMEAN(I)=0.
374 DO 30 J=1,N
      30 R(J) = XMEAN(I)
C
C      IDENTIFICATION LOOP (IDENTIFY THE PREDICTOR COEFFICIENTS)
C
C
      DO 100 J=1,NSF
      S = XV(J)
      S2 = 0.0
      RT = 0.0
      DO 40 K=1,N
      V1(K) = 0.
      DO 40 L=1,N
40  V1(K) = V1(K) + V(K,L)*R(L)
      DO 50 K=1,N
      S2 = S2 + R(K)*V1(K)
      DO 60 K=1,N
      G(K) = V1(K)/(VARI + S2)
60  RT = RT + A(K)*R(K)
      DO 70 K=1,N
      DO 70 L=1,K
      V(K,L) = V(K,L) - G(K)*V1(L)
70  V(L,K) = V(K,L)
      E = S - RT
      DO 80 K=1,N
80  A(K) = A(K) + G(K)*E
C
C      SHIFT THE PAST VALUE VECTOR
C
      DO 90 L=2,N
      90 R(N+2-L) = R(N+1-L)
      R(1) = RT+E
100 CONTINUE
      CALL PARSBL(A,N,PAR,B)
      DO 335 KK=1,N
      CM(KK,I)=UNIFRM(-1.75,1.75,256,A(KK))
335 CONTINUE
C

```

*** Subroutine for ADPCM / Kalman filter / Zeroing ***

C RESET R-REGISTER TO INITIAL SIGNAL VALUE

C

```
DO 120 J=1,N
120 R(J) = XMEAN(I)
DO 150 J=1,NSF
S = XV(J)
RT2 = 0.0
```

C

C

DETERMINE RESIDUAL SIGNAL

C

C

RT2 IS AT THE TRANSMITTER STAGE WHERE:

C

C

RT2 = RT2 + AO(K)*RV(K)

C

```
DO 130 K=1,N
130 RT2 = RT2 + CM(K,I)*R(K)
E2V(J)=S-RT2
```

C

C

SHIFT R-REGISTER

C

```
DO 140 K=2,N
140 R(N+2-K) = R(N+1-K)
R(1) = RT2+E2V(J)
150 CONTINUE
SUME=0.
SUMSQE=0.
DO 155 M=1,NSF
SUME=SUME+E2V(M)
155 SUMSQE=SUMSQE+(E2V(M)*E2V(M))
VAREST(I)=(SUMSQE-(SUME*SUME/NSF))/NSFM1
IF(VAREST(I) .LT. VLO)VLO=VAREST(I)
IF(VAREST(I) .GT. VHIGH)VHIGH=VAREST(I)
VSME=VSME+VAREST(I)
VSMSE=VSMSE+VAREST(I)*VAREST(I)
RESMN(I)=SUME/NSF
IF(MOD(I-1,JBLKSZ) .EQ. 0)RESMN(I)=RESMN(I)/JBLKSZ
WRITE(6,919)I,RESMN(I),VAREST(I),(A(K1),K1=1,N),(CM(K2,I),K2=1,N)
919 FORMAT(' ',I4,5(1X,E10.3),/,3(1X,E10.3))
VAREST(I)=UNIFRM(0.,40000.,512,VAREST(I))
RSME=RSME+RESMN(I)
RSMSE=RSMSE+RESMN(I)*RESMN(I)
RESMN(I)=QNTZ(RESMN(I),MLEVEL,QML,QMTH,150.)
IF(MOD(I-1,JBLKSZ) .EQ. 0)RESMN(I)=RESMN(I)*JBLKSZ
DO 200 J=1,N
200 R(J) = XMEAN(I)
DO 250 J=1,NSF
S=XV(J)
RT2=0.0
```

C

C

NOW GET XMITTER RESIDUAL

C

*** Subroutine for ADPCM / Kalman filter / Zeroing ***

```

DO 230 K=1,N
230 RT2=RT2 + CM(K,I)*R(K)
  E2 = S - RT2 - RESMN(I)
  EQ2=QNTZ(E2,NLEVEL,QL,QTH,VAREST(I))
  IF(IBEAT.NE.1)EQ2=0.
  IF(IBEAT.EQ.LIMCNT)IBEAT=0
  IBEAT=IBEAT+1
  XTR(I,J)=EQ2
C
C
C  SHIFT R-REGISTER
C
C
DO 240 K=2,N
240 R(N+2-K) = R(N+1-K)
  R(1) = RT2+EQ2+RESMN(I)
250 CONTINUE
160 CONTINUE
  VMN=VSME/256.
  VVAR=(VSMSE-((VSME*VSME)/256.))/255.
  RMN=RSME/256.
  RVAR=(RSMSE-((RSME*RSME)/256.))/255.
  WRITE(6,920)VMN,VVAR,RMN,RVAR
920 FORMAT(' VMN=',E12.5,' VVAR=',E12.5,' RMN=',E12.5,' RVAR=',E12.5)
  WRITE(6,921)VLO,VHIGH
921 FORMAT(' VLO=',E15.5,' VHIGH=',E15.5)
C-----
C
C  RECEIVER SIMULATION SECTION
C-----
C
C  SET R - REGISTER TO MEAN VALUE
C  FOR THE CURRENT LINE.
C
C
DO 210 I = 1,NF
DO 170 J = 1,N
170 R(J)=XMEAN(I)
  DO 400 J = 1,NSF
  S = XIN(I,J)
  RV2 = 0.0
  DO 180 K = 1,N
180 RV2 = RV2 + CM(K,I)*R(K)
  RN = XTR(I,J) + RV2 + RESMN(I)
  XIN(I,J) = RN
C-----
C
C  SHIFT R - REGISTER.
C

```

*** Subroutine for ADPCM / Kalman filter / Zeroing ***

C

```
      DO 190 K = 2,N
190   R(N+2-K) = R(N+1-K)
      R(1) = RN
400   CONTINUE
210   CONTINUE
      RETURN
      END
```

*** Subroutine for ADPCM / Kalman filter / Back-Quantization ***

```
SUBROUTINE COMPMC(XIN,NF,NSF,N,CM,QTH,QL,NLEVEL)
REAL A(10),V(10),XIN(256,256),R(10),XV(256),G(10),V(10,10)
REAL XTR(256,256)
REAL CM(7,256),QL(1),QTH(1),E2V(256),XMEAN(256)
REAL RESMN(256)
REAL QML(1024),QMT(1024)
REAL B(10,10),PAR(10)
```

EXPLANATION OF CALL VARIABLES:

```
XIN    - THE INPUT DATA MATRIX AND RESIDUAL OUTPUT MATRIX
NF     - NUMBER OF FRAMES (NUMBER OF LINES IN THE INPUT MATRIX)
NSF    - NUMBER OF SAMPLES PER FRAME (SAMPLES PER LINE)
N      - NUMBER OF PREDICTOR COEFFICIENTS TO BE USED
CM     - MATRIX THAT CONTAINS THE PREDICTOR COEFFICIENT VECTORS
```

DEFINITION OF VARIABLE TERMS:

```
V      - THE ERROR COVARIANCE MATRIX
A      - THE PREDICTOR COEFFICIENT VECTOR
VARI   - INITIAL VALUE FOR THE ERROR COVARIANCE MATRIX
VV     - VARIANCE OFFSET
XV     - THE INPUT LINE TEMPORARY VECTOR
G      - THE GAIN VECTOR
R      - THE PAST VALUE VECTOR
E      - THE ERROR OR RESIDUAL TERM
```

SET UP THE CONSTANTS AND INITIAL VALUES

```
DATA V/100*0.0/,A/1.0,-.5,-.2,.3,.4,-.5/
CALL MAXQTZ(QMT,QML,MLEVEL,6)
VLO=9999999.
VHIGH=-1.
XNSF = NSF
COMMON /A/ JBLKSZ
XAVBLK=JBLKSZ*128.
NSFM1=NSF-1
VV = 1.0
```

IN THE DO 160 LOOP, I IS THE LINE NUMBER (1 - NF)

```
DO 160 I=1,NF
VARI=100.
DO 10 J=1,N
DO 10 K=1,N
V(J,K) = 0.0
IF(J.EQ.K) V(J,K) = VARI
10 CONTINUE
```

*** Subroutine for ADPCM / Kalman filter / Back-Quantization ***

C SET UP THE INPUT VECTOR AND THE PAST VALUE VECTOR

C

```

SUMX=0.
DO 20 J=1,NSF
SUMX=SUMX+XIN(I,J)
20  XV(J) = XIN(I,J)
XMEAN(I)=SUMX/XNSF
WRITE(6,2000) I,XMEAN(I)
2000 FORMAT(' LINE: ',I5,2X,E15.5)
IF(MOD(I-1,JBLKSZ) .NE. 0)GO TO 373
XMEAN(I)=2750.
GO TO 374
373 XMEAN(I)=0.
374 DO 30 J=1,N
30  R(J) = XMEAN(I)

```

C

C . IDENTIFICATION LOOP (IDENTIFY THE PREDICTOR COEFFICIENTS)

C

C

```

DO 100 J=1,NSF
S = XV(J)
S2 = 0.0
RT = 0.0
DO 40 K=1,N
V1(K) = 0.
DO 40 L=1,N
40  V1(K) = V1(K) + V(K,L)*R(L)
DO 50 K=1,N
50  S2 = S2 + R(K)*V1(K)
DO 60 K=1,N
G(K) = V1(K)/(VARI + S2)
60  RT = RT + A(K)*R(K)
DO 70 K=1,N
DO 70 L=1,K
V(K,L) = V(K,L) - G(K)*V1(L)
70  V(L,K) = V(K,L)
E = S - RT
DO 80 K=1,N
80  A(K) = A(K) + G(K)*E

```

C

C

C

SHIFT THE PAST VALUE VECTOR

```

DO 90 L=2,N
90  R(N+2-L) = R(N+1-L)
R(1) = RT+E
100 CONTINUE
CALL PARSBL(A,N,PAR,B)
DO 335 KK=1,N
CM(KK,I)=UNIFRM(-1.75,1.75,256,A(KK))
335 CONTINUE

```

*** Subroutine for ADPCM / Kalman filter / Back-Quantization ***

```

C
C      RESET R-REGISTER TO INITIAL SIGNAL VALUE
C
      DO 120 J=1,N
120  R(J) = XMEAN(I)
      DO 150 J=1,NSF
      S = XV(J)
      RT2 = 0.0

C
C      DETERMINE RESIDUAL SIGNAL
C
C      RT2 IS AT THE TRANSMITTER STAGE WHERE:
C      RT2 = RT2 + AO(K)*RV(K)
C
      DO 130 K=1,N
130  RT2 = RT2 + CM(K,I)*R(K)
      E2V(J)=S-RT2

C
C      SHIFT R-REGISTER
C
      DO 140 K=2,N
140  R(N+2-K) = R(N+1-K)
      R(1) = RT2+E2V(J)
150  CONTINUE
      SUME=0.
      SUMSQE=0.
      DO 155 M=1,NSF
      SUME=SUME+E2V(M)
155  SUMSQE=SUMSQE+(E2V(M)*E2V(M))
      RESMN(I)=SUME/XNSF
      IF(MOD(I-1,JBLKSZ) .EQ. 0) RESMN(I)=RESMN(I)/JBLKSZ
      WRITE(6,919) I, RESMN(I), (A(K1),K1=1,N), (CM(K2,I),K2=1,N)
919  FORMAT(' ',I4,4(1X,E10.3),/,3(1X,E10.3))
      RSME=RSME+RESMN(I)
      RSMSE=RSMSE+RESMN(I)*RESMN(I)
      RESMN(I)=QNTZ(RESMN(I),MLEVEL,QML,QMTH,150.)
      IF(MOD(I-1,JBLKSZ) .EQ. 0) RESMN(I)=RESMN(I)*JBLKSZ
      DO 200 J=1,N
200  R(J) = XMEAN(I)
      DELTA=100.
      ALPHA1=.7
      ALPHA2=1.
      RESMSQ=0.
      DO 250 J=1,NSF
      S=XV(J)
      RT2=0.0

C
C      NOW GET XMITTER RESIDUAL
C
      DO 230 K=1,N

```

*** Subroutine for ADPCM / Kalman filter / Back-Quantization ***

```

230 RT2=RT2 + CM(K,I)*R(K)
    E2 = S - RT2 - RESMN(I)
    EQ2=QNTZ(E2,NLEVEL,QL,QTH,DELTA)
    XTR(I,J)=EQ2
    E2V(J)=EQ2
    DELTA=0.
    DO 701 JI=1,J
701 DELTA=DELTA+(ALPHA1** (JI-1))*ABS(E2V(J-JI+1))
    DELTA=((1-ALPHA1)/ALPHA2)*DELTA
    DELTA=DELTA*DELTA
    IF(DELTA .LT. VLO) VLO=DELTA
    IF(DELTA .GT. VHIGH) VHIGH=DELTA

C
C
C SHIFT R-REGISTER
C
C
    DO 240 K=2,N
240 R(N+2-K) = R(N+1-K)
    R(1) = RT2+EQ2+RESMN(I)
250 CONTINUE
160 CONTINUE
    VMN=VSME/256.
    VVAR=(VSMSE-((VSME*VSME)/256.))/255.
    RMN=RSME/256.
    RVAR=(RSMSE-((RSME*RSME)/256.))/255.
    WRITE(6,920) VMN,VVAR,RMN,RVAR
920 FORMAT(' VMN=',E12.5,' VVAR=',E12.5,' RMN=',E12.5,' RVAR=',E12.5)
    WRITE(6,921) VLO,VHIGH
921 FORMAT(' VLO=',E15.5,' VHIGH=',E15.5)
C-----
C
C RECEIVER SIMULATION SECTION
C-----
C
C SET R - REGISTER TO MEAN VALUE
C FOR THE CURRENT LINE.
C
C
    DO 210 I = 1,NF
    DO 170 J = 1,N
170 R(J)=XMEAN(I)
    DO 400 J = 1,NSF
    S = XIN(I,J)
    RV2 = 0.0
    DO 180 K = 1,N
180 RV2 = RV2 + CM(K,I)*R(K)
    RN = XTR(I,J) + RV2 + RESMN(I)
    XIN(I,J) = RN

```

*** Subroutine for ADPCM / Kalman filter / Back-Quantization ***

C-----

C

C

C

C

C

SHIFT R - REGISTER.

DO 190 K = 2,N
190 R(N+2-K) = R(N+1-K)
R(1) = RN
400 CONTINUE
210 CONTINUE
RETURN
END

*** Main Driver Program for VASPC ***

INTEGER IN2/4HIN*2/,IN4/4HIN*4/,RL/4HREAL/
 REAL*4 XIN(256,256),COEFF(7,256)
 REAL A(256)

C-----
 C
 C
 C
 C VAHPC MAIN DRIVER PROGRAM
 C
 C EXPLANATIONS OF VARIABLES:
 C XIN - MAIN I/O IMAGE MATRIX
 C COEFF - COEFFICIENT STORAGE MATRIX
 C JBLKSZ - TRANSFORMATION BLOCKSIZE
 C ISZ - IMAGE ROW SIZE
 C JSZ - IMAGE COLUMN SIZE
 C NUMSMP - NUMBER SAMPLES/LINE (SAME AS JSZ)
 C NUMFR - NUMBER SAMPLE LINES/FRAME (SAME AS ISZ)
 C LVECTR - NUMBER OF VECTORS IN CODE BOOK
 C KDIM - VECTOR DIMENSION
 C NCOEFF - NUMBER OF PREDICTOR COEFFICIENTS
 C
 C SUBROUTINES:
 C DSKIO - DISK INPUT / OUTPUT
 C COMPV - ADPCM DATA COMPRESSION
 C USING PARALLEL PROCESSING
 C AND VECTOR QUANTIZATION
 C
 C-----

NUMFR=256
 NUMSMP=256
 NCOEFF=3
 LVECTR=16
 LVECSQ=LVECTR*LVECTR
 LVECS1=LVECSQ-1
 KDIM=8
 ISZ=256
 JSZ=256
 JBLKSZ=16
 COMMON /A/ JBLKSZ
 DO 10 I=1,ISZ
 CALL DSKIO(A,JSZ,I,1,1,RL)
 DO 10 J=1,JSZ
 10 XIN(I,J)=A(J)
 DO 11 I=1,12
 11 WRITE(6,1001)(XIN(I,J),J=1,16)
 1001 FORMAT(' ',16(F6.0,1X))
 CALL COMPV(XIN,NUMFR,NUMSMP,NCOEFF,COEFF,'VECTR,KDIM)
 WRITE(6,20)
 20 FORMAT(' ***** MADE IT BACK THROUGH SUBS *****')
 DO 40 I=1,ISZ

*** Main Driver Program for VASPC ***

```
      DO 35 J=1,JSZ
35  A(J)=XIN(I,J)
      CALL DSKIO(A,JSZ,I,0,2,RL)
40  CONTINUE
      WRITE(6,2)
2   FORMAT('0*** JOB FINISHED ***')
      STOP
      END
```

*** Max-Lloyd Optimal Quantizer Design Subroutine ***

SUBROUTINE MAXQTZ (QTH,QL,N,NBIT)

C-----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

OPTIMAL QUANTIZER SUBROUTINE
THIS PROGRAM CALCULATES THE OPTIMUM QUANTIZATION
LEVELS FOR A SPECIFIC NUMBER OF LEVELS AS OUT-
LINED BY JOEL MAX.
THE PROBABILITY DENSITY FUNCTION IS ASSUMED SYM-
METRIC AND IS DEFINED AS A FUNCTION, HERE AS
FUNC.NORMAL, AND IS ACCESSED THROUGH "FUNC".

REAL*8 X(1024),Y(1024),AREA,XLAST,THELST,XB(1024),YB(1024)
REAL*4 QTH(1),QL(1),ERROR
REAL*8 EXTRA,ADD
N=2**NBIT
WRITE(6,3)N
3 FORMAT('0NUM LEVELS:',I5,/)

THIS SECTION ZEROES OUT ALL ARRAYS AND SETS UP
ALL INITIAL CRITERION ESTIMATES.

DO 5 K=1,1024
X(K)=0.
XB(K)=0.
YB(K)=0.
5 Y(K)=0.
XLAST=999.
ADD=.1
IF(N .GT. 7)ADD=.01
IF(N .GT. 20)ADD=.0005
IF(N .GT. 128)ADD=5.D-8
GUESS=0.0001
TYPE=0.

THIS SECTION FIGURES OUT IF N IS ODD OR EVEN.

TYPE=1. --> ODD
TYPE=0. --> EVEN

CHECK=N/2.
ISTOP=N/2
IF((CHECK-FLOAT(INT(CHECK))) .NE. 0.)TYPE=1.
IF(TYPE .NE. 1.)GO TO 10

*** Max-Lloyd Optimal Quantizer Design Subroutine ***

```

C-----
C
C THIS SECTION SETS UP INITIAL ESTIMATES OF OPTIMUM
C SIGNAL LEVELS (X), AND QUANTIZATION LEVELS (Y).
C
C

```

```

      Y(1)=0.
      X(1)=GUESS
      THELST=X(1)
      Y(2)=2*X(1)-Y(1)
      GO TO 20
10  X(1)=0.
      Y(1)=GUESS
      THELST=Y(1)

```

```

C-----
C
C WE NOW CALCULATE ALL SUCCEEDING SIGNAL AND
C QUANTIZATION LEVELS AS OUTLINED BY MAX.
C
C

```

```

20  DO 50 J=1,ISTOP
      I=J+1
      JJ=J
      IF(TYPE .EQ. 1.)JJ=J+1
18  CALL CENTRD(X(J),X(I),Y(JJ),AREA,0.)
50  Y(JJ+1)=2*X(I)-Y(JJ)

```

```

C-----
C
C WE NOW SEE IF THE LAST Y-LEVEL (QUANTIZATION) IS
C THE CENTROID BETWEEN THE PREVIOUS X-LEVEL
C (SIGNAL) AND INFINITY.
C
C

```

```

      JTRAN=JJ+1
      IEND=ISTOP
      IF(TYPE .EQ. 1)IEND=ISTOP+1
      IF(N .LE. 128)CALL CENTRD(X(ISTOP),15.,Y(IEND),AREA,1.)
      IF(N .GT. 128)CALL CENTRD(X(ISTOP),100.,Y(IEND),AREA,1.)
      IF(N .GT. 128)GO TO 55
      IF((ADD .LE. 1.D-6) .OR. (DABS(AREA) .LE. 1.D-6))GO TO 80
      GO TO 56
55  IF((ADD .LE. 1.D-10) .OR. (DABS(AREA) .LE. 1.D-6))GO TO 80

```

```

C-----
C
C IS THE /AREA/ WITH THESE VALUES LESS THAN THE
C LAST ITERATION?
C
C IF SO, CHECK TO SEE HOW CLOSE AREA IS TO
C ZERO, AND IF NECESSARY, MODIFY INITIAL
C ESTIMATES AND ITERATE AGAIN.

```

*** Max-Lloyd Optimal Quantizer Design Subroutine ***

```

C
C   IF NOT, RESET TO LAST ITERATION, AND REDUCE
C   THE MODIFICATION VARIABLE (ADD).
C
C
56 IF(DABS(AREA) .LE. DABS(XLAST))GO TO 60
   IF(TYPE .NE. 1.)GO TO 57
   X(1)=THELST
   IF(XLAST .EQ. 999.)GO TO 58
   ADD=ADD/2.
   GO TO 58
57 Y(1)=THELST
   IF(XLAST .EQ. 999.)GO TO 58
   ADD=ADD/2.
58 DO 59 J=1,JTRAN
   X(J)=XB(J)
59 Y(J)=YB(J)
60 XLAST=AREA
   IF(TYPE .NE. 1.)GO TO 65
   THELST=X(1)
   GO TO 67
65 THELST=Y(1)
67 IF(AREA .LT. 0.)GO TO 68
   EXTRA=ADD
   GO TO 69
68 EXTRA=-ADD
69 DO 70 J=1,JTRAN
   XB(J)=X(J)
70 YB(J)=Y(J)
   IF(TYPE .NE. 1.)GO TO 75
   X(1)=X(1)+EXTRA
   Y(2)=2*X(1)
   GO TO 20
75 Y(1)=Y(1)+EXTRA
   GO TO 20
-----
C
C   WE NOW WRITE OUT THE CALCULATED OUTPUT LEVELS
C   -- SIGNAL LEVELS AND CORRESPONDING QUANTIZER
C   LEVELS.
C
C
80 NQL=N
   NQT=N-1
   IF (TYPE .NE. 1.)GO TO 300
   NX=INT(N/2.)
   NY=INT(N/2.)+1
   FLG=0.
   IPT=NX
   DO 205 I=1,NQT

```

*** Max-Lloyd Optimal Quantizer Design Subroutine ***

```

      XXX=-X(IPT)
      IF (FLG .EQ. 1.)GO TO 201
      QTH(I)=-X(IPT)
      IF (FLG .EQ. 0.)GO TO 202
      I=I+1
201  QTH(I)=X(IPT)
      IPT=IPT+1
      GO TO 205
202  IF (IPT .EQ. 1)FLG=1.
      IF (IPT .NE. 1)IPT=IPT-1
205  CONTINUE
      FLG=0.
      IPT=NY
      DO 210 I=1,NQL
      IF (IPT .EQ. 1)FLG=1.
      IF (FLG .EQ. 1.)GO TO 206
      QL(I)=-Y(IPT)
      YYY=-Y(IPT)
      IPT=IPT-1
      GO TO 210
206  QL(I)=Y(IPT)
      IPT=IPT+1
210  CONTINUE
      GO TO 999
300  NX=INT(N/2.)
      NY=INT(N/2.)
      FLG=0.
      IPT=NX
      DO 305 I=1,NQT
      IF (IPT .EQ. 1)FLG=1.
      IF (FLG .EQ. 1.)GO TO 301
      QTH(I)=-X(IPT)
      IPT=IPT-1
      GO TO 305
301  QTH(I)=X(IPT)
      IPT=IPT+1
305  CONTINUE
      FLG=0.
      IPT=NY
      DO 310 I=1,NQL
      IF (FLG .EQ. 1.)GO TO 306
      QL(I)=-Y(IPT)
      IF (FLG .EQ. 0.)GO TO 307
      I=I+1
306  QL(I)=Y(IPT)
      IPT=IPT+1
      GO TO 310
307  IF (IPT .EQ. 1)FLG=1.
      IF (IPT .NE. 1)IPT=IPT-1
310  CONTINUE

```

*** Max-Lloyd Optimal Quantizer Design Subroutine ***

C
C

999 CONTINUE
RETURN
END

*** Functions for Probability Calculations ***

FUNCTION FUNC(X,Q)

C-----

C

C

C

C

NORMAL GAUSSIAN DENSITY FUNCTION

REAL*8 FUNC,X,Q,P,ARG

ARG=-X*X/2.

IF(ARG.LT. -150.)GO TO 1

P=.3989422804*DEXP(ARG)

GO TO 2

1 P=0.

2 FUNC=(X-Q)*P

RETURN

END

FUNCTION SQERR(X,Q)

C-----

C

C

C

C

C

C

MEAN SQUARED ERROR FUNCTION FOR
THE NORMAL GAUSSIAN DENSITY.

REAL*4 SQERR,X,Q,P

P=.3989422804*EXP(-X*X/2.)

SQERR=(X-Q)*(X-Q)*P

RETURN

END

*** Function for Adaptive Quantization ***

FUNCTION QNTZ(X,L,QL,QTH,VAR)

C-----
C
C ADAPTIVE QUANTIZER FUNCTION WHICH
C QUANTIZES INPUT SAMPLES GIVEN THE
C DESIRED LEVELS AND THRESHOLDS ALONG
C WITH THE VARIANCE ESTIMATE.
C
C

REAL QL(1),QTH(1)
LM1=L-1
STD=SQRT(VAR)
DO 10 K=1,LM1
THRESH=QTH(K)*STD
IF(X .GE. THRESH)GO TO 10
QNTZ=QL(K)*STD
RETURN
10 CONTINUE
QNTZ=QL(L)*STD
RETURN
END

*** Function for Uniform Quantization ***

```
FUNCTION UNIFRM(SMALL,BIG,N,X)
  IF(X .LE. SMALL)GO TO 20
  IF(X .GE. BIG)GO TO 40
  NM1=N-1
  ADD=(BIG-SMALL)/NM1
  THRADD=ADD/2.
  THRSH2=SMALL
  DO 10 I=1,NM1
  THRSH1=THRSH2
  DECIDE=THRSH1+THRADD
  THRSH2=THRSH1+ADD
  IF((X.GT.THRSH1) .AND. (X.LE.DECIDE))GO TO 11
  IF((X.GT.DECIDE) .AND. (X.LE.THRSH2))GO TO 12
10 CONTINUE
  WRITE(6,1)
  1 FORMAT(' *.*.*.*>> UNIFORM LOGIC ERROR <<*.*.*.*')
11 UNIFRM=THRSH1
  GO TO 99
12 UNIFRM=THRSH2
  GO TO 99
20 UNIFRM=SMALL
  GO TO 99
40 UNIFRM=BIG
99 RETURN
END
```

*** Subroutine for Centroid Calculations ***

SUBROUTINE CENTRD(XI,XIPL,Y,AREA,FLAG)

CENTRD IS A SUBROUTINE DESIGNED TO:

- 1) CALCULATE THE CENTROID PIVOT ELEMENT BETWEEN TWO ENDPOINTS OF A KNOWN PROBABILITY DENSITY FUNCTION.
- 2) CALCULATE THE AREA WITH Y AS A SPECIFIC CENTROID PIVOT ELEMENT.

XI, XIPL ARE THE INTEGRAL ENDPOINTS.
Y IS THE PIVOTAL ELEMENT.

$$\text{AREA} = \int_{\text{XI}}^{\text{XIPL}} (X-Y) * P(X) \text{ DX}$$

FUNCTION 1) IS SELECTED IF FLAG<>1.
FUNCTION 2) IS SELECTED IF FLAG=1.

```

REAL*8 XI,XIPL,Y,AREA,YL,XLAST,ADTNL
N=0
IF(FLAG .NE. 1.)GO TO 5
CALL INTGRL(XI,XIPL,Y,AREA,0.)
GO TO 30
5 XLAST=999.
ADTNL=.001
CRTRN=.001
XIPL=2.*Y-XI
XL=XIPL
10 CALL INTGRL(XI,XIPL,Y,AREA,0.)
N=N+1
IF(N .EQ. 500)GO TO 30
IF(CRTRN .LT. DABS(AREA))GO TO 15
CRTRN=CRTRN/10.
IF(CRTRN .LE. 1.E-7)GO TO 30
15 IF(DABS(AREA) .LE. DABS(XLAST))GO TO 20
XIPL=XL
ADTNL=ADTNL/2.
CALL INTGRL(XI,XIPL,Y,AREA,0.)
20 XLAST=AREA
XL=XIPL
IF(AREA .LT. 0.)GO TO 25
XIPL=XIPL-ADTNL

```

*** Subroutine for Centroid Calculations ***

GO TO 10
25 XIPI=XIPI+ADTNL
GO TO 10
30 RETURN
END

```
C-----
C
C      VECTOR QUANTIZER DESIGN SUBROUTINE
C
C      A = QUANTIZER CODEBOOK
C      X = TRAINING SET
C      DSTRN = DISTORTION BETWEEN ITH VECTOR OF X, AND
C              JTH CODEVECTOR OF A.
C      EPS = DISTORTION THRESHOLD
C      KDIM = VECTOR SPACE DIMENSION (BLOCK LENGTH)
C      DM = OVERALL AVERAGE DISTORTION AT MTH STAGE
C      P = MINIMUM DISTORTION PARTITION
C      N = NUMBER OF QUANTIZATION LEVELS
C      NUM = NUMBER OF TRAINING VECTORS
C      SMALL = LIST OF SMALLEST DISTORTIONS
C      IPT = IDENTIFIES PARTITION SET FOR EACH
C            TRAINING SET.
C      NUMP = NUMBER OF TRAINING VECTORS IN
C             EACH PARTITION.
C
C      DIMENSIONALITY:
C          A(N,KDIM), X(KDIM,NUM),
C          DSTRN(NUM,N), SMALL(NUM), IPT(NUM),
C          NUMP(N)
C
C
C
C
C
C
C
C
C
C
C
C      REAL*4 A(64,16),X(KDIM,1),SEQNCE(1)
C      REAL*4 SMALL(2048),ATEMP(32,16),EPSLON(16)
C      INTEGER IPT(2048),NUMP(16)
C      DATA EPSLON/16*.5/
C      DO 6 I=1,N
C      DO 6 K=1,KDIM
C 6   A(I,K)=0.
C      NUM=ISEQSZ/KDIM
C      M=0
C      EPS=0.0001
C      DLAST=1.E50
C      DO 1 I=1,NUM
C      DO 1 K=1,KDIM
C 1   A(1,K)=A(1,K)+X(K,I)/NUM
C      NCOUNT=1
C-----
C
C      SPLIT CODEBOOK
```

*** Subroutine for Vector Quantizer Design ***

C
C

```

5 DO 2 I=1,NCOUNT
  DO 2 K=1,KDIM
2  ATEMP(I,K)=A(I,K)
  ICNT=0
  DO 4 I=1,NCOUNT
    ICNT=ICNT+1
  DO 3 K=1,KDIM
3  A(ICNT,K)=ATEMP(I,K)+EPSLON(K)
  ICNT=ICNT+1
  DO 4 K=1,KDIM
4  A(ICNT,K)=ATEMP(I,K)-EPSLON(K)
  NCOUNT=2*NCOUNT
10 IF(M .EQ. 100)GO TO 999
  DO 200 I=1,NUM
    SMALL(I)=1.E50
    IPT(I)=1
    DO 100 J=1,NCOUNT
      DSTRN=0.
      DO 50 K=1,KDIM
50  DSTRN=DSTRN+(X(K,I)-A(J,K))*(X(K,I)-A(J,K))
      IF(DSTRN .GT. SMALL(I))GO TO 100
      SMALL(I)=DSTRN
      IPT(I)=J
100 CONTINUE
200 CONTINUE
  DO 300 J=1,NCOUNT
    NUMP(J)=0
    DO 300 I=1,NUM
      IF(IPT(I) .NE. J) GO TO 300
      NUMP(J)=NUMP(J)+1
300 CONTINUE
  DM=0.
  DO 400 I=1,NUM
400 DM=DM+SMALL(I)
  DM=DM/NUM
  WRITE(6,772)DM
772 FORMAT(' DM:',E15.5)
  IF(DM .EQ. DLAST) GO TO 999
  IF(ABS((DLAST-DM)/DM) .LE. EPS) GO TO 999
  DLAST=DM
  DO 500 I=1,NCOUNT
    IF(NUMP(I) .EQ. 0) GO TO 500
    NVECS=NUMP(I)
    DO 420 K=1,KDIM
420 A(I,K)=0.
    IF(NVECS .EQ. 0)GO TO 430
    DO 430 J=1,NUM
      IF(IPT(J) .NE. I)GO TO 430

```

AD-A138 876 ADAPTIVE HYBRID PICTURE CODING(U) ARKANSAS UNIV
FAVETTEVILLE DEPT OF ELECTRICAL ENGINEERING
R A JONES ET AL 30 NOV 83 AFOSR-TR-84-0142

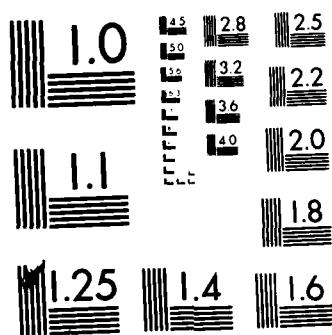
3/3

UNCLASSIFIED AFOSR-82-0351

F/G 12/1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

*** Subroutine for Vector Quantizer Design ***

```
      DO 425 K=1,KDIM
425  A(I,K)=A(I,K)+X(K,J)/NVECS
430  CONTINUE
500  CONTINUE
      M=M+1
      GO TO 10
999  IF(NCOUNT.NE.N)GO TO 5
      WRITE(6,888)M
888  FORMAT(' M=',I5)
      RETURN
      END
```

*** Subroutine for Vector Quantization ***

SUBROUTINE VECQNT(X,KDIM,N,CDBK)

C-----

C

C

C

C

C

C

VECTOR QUANTIZER:

EXHAUSTIVE SEARCH FOR MIN DISTORTION.

REAL*4 X(1),CDBK(64,16)

SMALL=1.E50

IPT=1

DO 50 J=1,N

DSTRN=0.

DO 25 K=1,KDIM

25 DSTRN=DSTRN+(X(K)-CDBK(J,K))*(X(K)-CDBK(J,K))

IF(DSTRN .GT. SMALL) GO TO 50

SMALL=DSTRN

IPT=J

50 CONTINUE

DO 75 K=1,KDIM

75 X(K)=CDBK(IPT,K)

RETURN

END

*** Program for Tape / Disk Transfersals ***

```
      INTEGER*2 IP(256)
      LOGICAL*1 PPICT(2,256)
      EQUIVALENCE(IP(1),PPICT(1,1))
      DO 999 N=1,6
      DO 10 I=1,256
      READ(N)IP
      DO 7 KK=1,256
      IF (IP(KK) .GT. 255) IP(KK)=255
      IF (IP(KK) .LT. 0) IP(KK)=0
7 CONTINUE
      WRITE(8,11) (PPICT(2,MM),MM=1,256)
11 FORMAT(4(64A1))
10 CONTINUE
      ENDFILE 8
999 CONTINUE
      REWIND 8
      STOP
      END
```

*** Program for SNR Calculations and Histogram Generation ***

```

INTEGER*2 R(256),T(256)
INTEGER*4 RI(256),TI(256),ORIG(256,256),TWOISZ,TWOIPl
DATA R/256*0/
REAL H(12,513),TMSE(12),SNR(12),RATIO(12),HWR(257)
REAL SIG(16)
DATA SIG/16*0./

```

```

C-----
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C

```

PROGRAM TO CALCULATE THE SNR RATIO OF AN IMAGE
AND GENERATE A HISTOGRAM OF THE ERROR BETWEEN
TWO IMAGES.

```

ISZ=256
READ(5,17) NUM
17 FORMAT(I2)
SQISZ=FLOAT(ISZ)*FLOAT(ISZ)
TWOISZ=2*ISZ
ISZPl=ISZ+1
TWOIPl=TWOISZ+1
INCRMT=512-ISZ
SUM=0.
DO 10 I=1,ISZ
  READ(1)T
  DO 10 J=1,ISZ
    TI(J)=T(J)
    SUM=SUM+FLOAT(TI(J))
10 ORIG(I,J)=TI(J)
  AVE=SUM/SQISZ
  SIG2=0.
  I=0
  IC=0
  DO 16 ID=1,ISZ,16
    IC=IC+1
    DO 15 IN=1,16
      I=I+1
      DO 15 J=1,ISZ
15 SIG(IC)=SIG(IC)+(FLOAT(ORIG(I,J))-AVE)*(FLOAT(ORIG(I,J))-AVE)
16 SIG2=SIG2+SIG(IC)
    DO 200 N=1,NUM
      NUMOFF=N+7
      DO 20 J=1,TWOISZ
20 H(N,J)=0.
    ERR2=0.
    WRITE(6,2)N
  2 FORMAT(' READING IMAGE #',I3)
  I=0

```

*** Program for SNR Calculations and Histogram Generation ***

```

IC=0
DO 50 IOUT=1,ISZ,16
IC=IC+1
ERR1=0.
DO 49 IN=1,16
I=I+1
READ(NUMOFF) R
DO 49 J=1,ISZ
RI(J)=R(J)
IDIFF=ORIG(I,J)-RI(J)
IF(IDIFF.LT.(ORIG(I,J)-255))IDIFF=ORIG(I,J)-255
IF(IDIFF.GT.ORIG(I,J))IDIFF=ORIG(I,J)
INDEX=IDIFF+ISZP1
H(N,INDEX)=H(N,INDEX)+1.0
49 ERR1=ERR1+FLOAT(IDIFF)*FLOAT(IDIFF)
ERR2=ERR2+ERR1
BLKSQ=4096.**2.
TMS1=ERR1/BLKSQ
SNR1=SIG(IC)/ERR1
RAT1=10.*ALOG10(SNR1)
WRITE(6,400) IC,TMS1,SNR1,RAT1
400 FORMAT(' BLK: ',I5,3(2X,E15.5))
50 CONTINUE
100 TMSE(N)=ERR2/SQISZ
SNR(N)=SIG2/ERR2
200 RATIO(N)=10.*ALOG10(SNR(N))
DO 300 N=1,NUM
DO 250 J=1,TWOIP1
IF((J.LT.128).OR.(J.GT.384))GO TO 250
JPT=J-127
HWR(JPT)=H(N,J)
250 CONTINUE
WRITE(24) HWR
WRITE(6,1) N, TMSE(N), SNR(N), RATIO(N)
1 FORMAT(' #',I3,' MSE: ',E15.5,' SNR: ',E15.5,' RATIO: ',E15.5)
300 CONTINUE
STOP
END

```

*** Subroutine to Plot Histograms -- CALCOMP ***

```

      REAL HWR(257),H(259),X(259)
      IB=1
      NC=1
      CALL PLOTS (IB,NC,25)
      CALL PLOT (1.75,1.0,-3)
      LG=257
      NUM=6
      DO 500 LOOP=1,NUM
      WRITE(6,11)
11  FORMAT(' /*CHANGE PAPER THEN ENTER: 1')
      READ(5,12)ANS
12  FORMAT(1A1)
      READ(24)HWR
      WRITE(25,13)
13  FORMAT(' !COPY HO PL1')
      DO 10 I=1,LG
10  H(I)=HWR(I)
      DO 50 I=1,LG
50  X(I)=I-129
      X(LG+1)=-128.0
      X(LG+2)=32.0
      H(LG+1)=0.0
      H(LG+2)=2200.0
      CALL AXIS (0.,0.,'PIXEL ERROR',-11,8.0,0.0,X(LG+1),X(LG+2))
      CALL AXIS (0.,0.,'FREQUENCY ',10,5.0,90.,H(LG+1),H(LG+2))
      DO 100 I=1,LG,1
      XX=(X(I)-X(LG+1))/X(LG+2)
      YY=H(I)/H(LG+2)
      CALL PLOT(XX,YY,3)
100 CALL PLOT(XX,0.,2)
      CALL SYMBOL(1.5,5.0,0.21,'HISTOGRAM OF THE ERROR',0.0,22)
      CALL SYMBOL(2.55,4.75,0.21,'FOR THE CITY',0.0,12)
      CALL SYMBOL(3.075,4.5,0.21,'256X256',0.0,7)
      CALL PLOT(12.0,9.0,999)
500 CONTINUE
      STOP
      END
      SUBROUTINE PSPCHR
C  SUB FOR SPECIAL CHARACTERS.
      RETURN
      END

```

END

FILMED

4-84

DTIC